Development of a novel tree search algorithm for reaction vector-based de novo design

By:

James Webster

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

This work was sponsored by

evotec

and

Biotechnology and Biological Sciences Research Council

The University of Sheffield

Faculty of Social Sciences

Information School

September 2021

Acknowledgements

Firstly, I would like to thank my supervisors Prof. Val Gillet, Prof. Beining Chen, and Dr. Mike Bodkin. Val for her tireless patience, guidance, advice, and support throughout the project. Beining, for her lively discussions and insights into medicinal chemistry and Mike for his industrial experience and tireless enthusiasm for drug discovery.

Next, I'd like to thank the "Reaction Vector Cowboys", Dr. Gian Marco Ghiandoni and Dr. James Wallace for the hours of discussions on reaction vectors and de novo design. Dr. Dimitar Hristozov for his industrial chemoinformatics knowledge and development of the reaction vector framework. Moreover, I'd like to thank my colleagues in the Sheffield chemoinformatics research group and the University of Sheffield; Peter, Antonio, Zied, Lucy, Christina, Matt, Moritz, Hanz, James, Nahal and Arshnous. Additionally, special thanks go to Jess Stacey, who started her journey in the chemoinformatics research group at the same time as me and has supported me throughout these last four years.

Finally, I'd like to thank my parents for their unyielding support and for always believing in my dream of becoming a scientist.

This work was made possible with the financial support of the BBSRC & Evotec.

Abstract

De novo design is a subdiscipline of chemoinformatics concerned with designing novel chemical entities in-silico that fit a pre-defined property profile. A fundamental limitation of de novo design is creating molecules that chemists can readily synthesise. Reaction vector-based de novo design aims to overcome this limitation by utilising reaction transformations mined from chemistry literature to build molecules of interest. However, de novo design utilising reaction vectors has been limited due to the reliance on complete enumeration, leading to a combinatorial explosion of products. Moreover, issues with local optima along sequential paths of reaction vectors have led to poor performance.

This project details the development of a novel tree search algorithm utilising reaction vectors. The Monte Carlo tree search (MCTS) was first identified as a promising approach to overcoming the limitations of reaction vector-based de novo design. The MCTS approach was implemented utilising the reaction vector methodology. To test the approach, a validated benchmark was constructed. The reaction vector-based Monte Carlo tree search (RVMCTS) performance was then investigated via a series of ablation studies utilising the validated benchmark. The studies revealed several limitations of the approach, which provided fruitful avenues for further modification.

First, the simulation component of the RVMCTS approach was investigated and modified by leveraging machine learning techniques. A further series of modifications were then investigated, including improved tree traversal and reaction vector prioritisation utilising reinforcement learning. These modifications were screened against the validated benchmark and combined to form REACTS (REACtion Tree Search), a novel tree search algorithm. Finally, REACTS was tested against a series of de novo design problems from the literature. REACTS demonstrated comparable performance against established deep learning-based approaches. Thus, REACTS demonstrated it can effectively chart a path through chemical space utilising the sequential application of reaction vectors to regions of interest.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

The design and creation of novel molecules drives the pharmaceutical industry. With the rapid adoption of computer systems and the birth of chemoinformatics, computer-aided drug design has become a cornerstone of many drug discovery campaigns in the pharmaceutical industry (Sliwoski et al., 2014). A sub-discipline of chemoinformatics is de novo design (Willett, 2008). De novo design is concerned with designing molecules from scratch. Early approaches established in the 1980s and 1990s struggled to build molecules that could be easily synthesisable in the lab (Schneider & Fechner, 2005). With the improvement of computers, more molecules could be generated; however, the resulting generated molecules remained a synthetic challenge. Therefore, an active research direction has been using reaction-based de novo design approaches to build up molecules in-silico via synthetic reaction transformations acquired from the chemistry literature (Vinkers et al., 2003). A prominent reaction-based de novo design approach is the reaction vector (RV) methodology (Patel, 2009; Patel et al., 2009). RVs were created for storage and search of chemical reactions in computer systems (Broughton et al., 2003). They were then applied to structure generation which is an ongoing area of research (Ghiandoni, 2019; Ghiandoni et al., 2019, 2020; Gillet et al., 2013; Hristozov et al., 2011; Wallace, 2016). However, the existing approaches to RV-based de novo design have two main limitations. First, RV-based de novo design relies on a full enumeration of all applicable RVs and reagents. Second, intermediate molecules may score poorly using the final design objective scores. The first limitation prevents longer synthetic routes, and thus more complex molecules, being designed due to a combinatorial explosion of products. The second limitation leads to molecules that score poorly but could lead to promising compounds being ignored during a search. Therefore, this thesis aims to overcome these issues via the use of reinforcement learning as part of a tree search algorithm.

Chapter 2 is an overview of molecular representation, including linear notations, non-linear notations, and storage of molecules on computers. Additionally, an overview of reaction storage, classification and a description of synthesis planning is provided.

Chapter 3 presents de novo design and the three core components required of a de novo design software. First, molecular scoring is reviewed, followed by molecular construction and finally, search and optimisation. The final section is dedicated to generative methods such as those inspired by text generation methods in natural language processing and deep learning.

Chapter 4 describes RVs and overviews their creation and usage as part of structure generation. This chapter will review the original implementation of RVs (Patel, 2009) and a revised algorithm approach (Hristozov et al., 2011).

Chapter 5 overviews machine learning and covers the algorithms and methods used to build machine learning models. Furthermore, this chapter also overviews machine learning on large datasets and covers reinforcement learning, a popular subset of machine learning dedicated to improving algorithms by interaction with an environment.

Chapter 6 introduces the Monte Carlo tree search (MCTS) algorithm, a multi-step reinforcement learning algorithm initially designed for generalised game playing (GGP) and its suitability for reaction-based de novo design. The MCTS algorithm is made of four components, selection, expansion, simulation, and backpropagation. An implementation of reaction vector-based MCTS (RVMCTS) is then described. Additionally, the chapter also describes the selection and creation of a validated benchmark used as part of this thesis.

Chapter 7 then explores the performance of each component of the RVMCTS. In particular, the study utilises an ablation study approach. By leveraging the ablation study approach, each RVMCTS component is investigated systematically.

Chapter 8 further explores the simulation component of the RVMCTS approach. This chapter first explores the usage of random simulation and highlights the complexities of the approach. Finally, the chapter introduces a machine learning approach to reduce the computational expense of simulation.

Chapter 9 modifies the RVMCTS with a variety of modifications. A single step reinforcement learning algorithm approach is introduced which utilises contextual bandits. Modifications are then tested, and the best performing modifications are identified. These are then combined to form REACTS which then demonstrated to solve problems from the validated benchmark.

Chapter 10 explores the use of REACTS on several de novo design problems. Three benchmark problems by (Brown et al., 2019) and four design problems explored previously by existing reaction-based de novo design software (Button et al., 2019). The results of the REACTS approach on the de novo design problems are presented.

Chapter 11 provides conclusions and limitations of the REACTS approach and overviews avenues for future work.

Chapter 2 Molecular and reaction representation

Molecules are physical entities made up of atoms and bonds. Molecules and collections of molecules have been studied and compiled in detail since ancient times. Moreover, in the last 100 years, with an increased output by chemists and the growth of computer systems, there was a growing need to represent molecules on computers for a wide range of tasks. Initially, these tasks were centred on information retrieval, where chemists would retrieve chemical information from databases (Baker et al., 1980; Weisgerber, 1997). However, as chemical information grew and the needs of users of chemical information systems became more complex, the field of chemoinformatics was born (Ray & Kirsch, 1957), although the term chemoinformatics was not coined until much later. As a result, chemoinformatics has grown to encompass various tasks, including structure search, molecular representation, the prediction of molecular and bioactive properties, reaction informatics, and compound design (Willett, 2011b).

This chapter first explores different approaches to molecular notations, including linear and non-linear notations and storing of molecules. Additionally, the first section of this chapter will then explore the usage of chemical graph theory and the chemical graph representation, focusing on applications such as structure search, substructure search, molecular descriptors, and similarity search.

The second part of this chapter will review reaction informatics and the principles of representing, retrieval, and classification of chemical reactions. Finally, applications of chemical reactions to the problem of synthesis planning will be presented.

2.1 Molecular representation

Chemical information systems require a machine-readable format that captures the essential information of a chemical structure. The International Union of Pure and Applied Chemistry (IUPAC) have a unique naming system that can convert a chemical structure to a name that can identify a molecule, an example of which is shown in Figure 2-1. However, the naming approach of IUPAC is cumbersome and does not allow the usage of common names (for example the common name of N-(4-hydroxyphenyl)acetamide is paracetamol). In addition to naming limitations, it may be tempting to represent a chemical structure as a 2D depiction, shown right in Figure 2-1. However, the depiction of a molecule that is an intuitive approach to describing chemistry for many chemists is also not suitable for storing a chemical structure.

Therefore, several notations, both linear and non-linear, have been developed to allow machine-readable and writeable descriptions of chemical structures. Several examples of linear notations are also shown in Figure 2-1. Moreover, non-linear notations have been developed to capture 3D

information of a chemical structure, an example of which is the grid of values at the bottom of Figure 2-1. Ultimately, depending on the chemical information required, different notations will be used.

Common names: Paracetamol, Acetoaminophen

IUPAC name: *N*-(4-hydroxyphenyl)ethanamide

Formula: $C_8H_9NO_2$

SMILES string: CC(=O)Nc1ccc(O)cc1

InChi/InChi key: 1S/C8H9NO2/c1-6(10)9-7-2-4-8(11)5-3-7/h2-5,11H,1H3,(H,9,10)

Mrv2003 09052118552D

```
 11 11  0  0  0  0            999 V2000
    0.0000    3.3000    0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    2.4750    0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
   -0.7145    2.0625    0.0000 O  0  0  0  0  0  0  0  0  0  0  0  0
    0.7145    2.0625    0.0000 N  0  0  0  0  0  0  0  0  0  0  0  0
    0.7145    1.2375    0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
    1.4289    0.8250    0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
    1.4289   -0.0000    0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
    0.7145   -0.4125    0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
    0.7145   -1.2375    0.0000 O  0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.8250    0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  2  0  0  0  0
  2  4  1  0  0  0  0
  4  5  1  0  0  0  0
  5  6  2  0  0  0  0
  6  7  1  0  0  0  0
  7  8  2  0  0  0  0
  8  9  1  0  0  0  0
  8 10  1  0  0  0  0
 10 11  2  0  0  0  0
  5 11  1  0  0  0  0
M  END
```

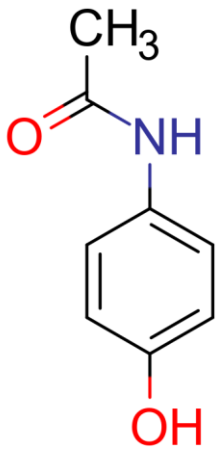


Figure 2-1 Paracetamol and corresponding notations

2.1.1 Linear notations

Linear notations describe a molecule in a single line of characters. Linear notations were first developed to overcome the limitations in memory that early chemical information systems possessed. One of the most popular linear notations from the earlier period of chemoinformatics was the Wiswesser Line-Formula Notation (WLN) (Wiswesser, 1952). WLN was later superseded by the, Simple Molecular Input Linear Entry System (SMILES) (Weininger, 1988). SMILES became more popular than WLN due to it being compact and human-readable, and it being possible to encode more complex chemical information than WLN (Warr, 2011). SMILES strings are a linear representation where the atoms and bonds are described based on walking a path across the molecule. A limitation of the SMILES approach is that generating a SMILES string does not yield unique representations. This limitation is often overcome by using established chemoinformatics software (*RDKit*, 2021) and a canonical approach to encoding the SMILES string (Weininger et al., 1989) whereby a unique ordering of the atoms is determined and a unique SMILES generated, described further below.

An alternative linear notation to the SMILES string is the INternational CHemical Identifier (InChI) (Heller et al., 2015). InChI is a linear notation designed by the InChI Trust to be a machine-readable writable format that uniquely describes a molecule. An important distinction is that the InChI notation was designed such that a single InChI is unambiguous. The InChI notation is a hierarchical identifier generated using software developed by the InChI Trust. The hierarchical approach is used to encode different chemical information. Moreover, the InChI hierarchical notation has limited use due to relying on the "\" character, leading to fragmentation of the character string making up the InChI on some computer systems. Hence, InChI keys are an alternative linear notation often used as a unique identifier. The InChI key is a hashed representation of the standard InChI (Heller et al., 2015) and does not contain the "\".

2.1.2 Non-Linear notations

Molecules are 3D entities, and chemists will often go to great lengths to determine the 3D structure of a molecule. Hence, an amenable alternative approach that allows the storing of 3D chemical data is via connection tables. Furthermore, with the advent of greater memory capacity, storing chemical structures as connection tables has become preferred (Warr, 2011).

Several different types of connection tables exist, however, the most common is the MDL connection table (Dalby et al., 1992). An MDL connection table is shown in Figure 2-1 and is comprised of blocks. Each block contains different information about the molecule. In MDL format, the atoms of the molecule and their coordinates are described in the atoms block. The bonds are described in the bonds block. As the atoms block allows coordinates to be stored this allows the MDL format to store 3D information. As a result, the MDL connection table format has grown in popularity. Several data file types use the MDL approach for molecule representation. These include SDF, which extend the MDL approach to include a data block that can capture additional metadata about the molecule.

2.1.3 Canonicalisation

An issue of both linear and non-linear molecular notations is that some notations such as SMILES are non-unique. To overcome this core limitation of chemical notations, chemical structures are often passed through canonicalisation processes. Canonicalisation is the process of acquiring a unique notation for a molecule. Canonicalisation approaches have been developed for both linear and non-linear notations. One of the main approaches to canonicalisation is via the Morgan algorithm (Morgan, 1965). The Morgan algorithm aims to overcome the limitations of notations by annotating each atom with a unique identifier. This identifier is then used as part of the generation of the unique notation, and thus, the notation becomes coupled to the annotated atoms.

The Morgan algorithm can be summarised using two steps. First, all atoms are labelled with a value of one. Then a recursive summation of the atom neighbours begins. The recursive summation adds all the atom neighbour values to the initial value of one for all atoms in the molecule. For example, if an atom in a molecule is connected to four single atoms, then the recursive summation would yield a value of five (four neighbours + itself). The recursive summation process repeats until the number of individual atom neighbour values does not increase. The atoms are now uniquely labelled, and canonicalised notation can be generated. Extensions to the Morgan algorithm have been developed to take into account stereochemistry (Wipke & Dyott, 1974). Furthermore, similar approaches are used to canonicalise SMILES strings (Weininger et al., 1989). Finally, while canonicalisation overcomes several limitations of generating molecular notations, canoncilisation does not reconcile that structures can exist in several states simultaneously (for example, through resonance and tautomerism) (Warr, 2011).

2.2 Chemical graph theory

So far, molecules have been treated as a set of notations. An alternative view is to represent molecules as a chemical graph. Graphs are mathematical structures composed of nodes and connected by edges. Molecules are composed of atoms connected by bonds. Therefore, a chemical graph is a graph where the atoms are the nodes, and the bonds are the edges. A benefit of the intuitive representation of a molecule as a chemical graph is that several algorithms and approaches first explored in mathematics can be readily applied to chemical problems (García-Domenech et al., 2008). One of the critical areas of development from chemical graph theory has been structure, substructure, and similarity search.

2.2.1 Structure search

Chemists often need to know if a compound is contained within a database. The first approach to structure searching is an exact match search. In graph theory, the problem of an exact match search is known as the graph isomorphism problem. An exact match search needs a query molecule and a list of molecules from the database to search through. As molecules are stored in databases using notations, it is possible to hash the notation using a hashing function (Freeland et al., 1979). A hash is an alphanumeric string obtained as output from a hashing function. Hashes have several beneficial properties, including sortability and minimisation of data redundancy (clashing). To search a database of molecules quickly, the list of molecule notations is converted to a list of hashes. The query molecule notation is also hashed. Finally, the query hash is compared to the list of database hashes. If one of these is the same as the query, then the record for the database hash is returned to the user.

An extension of exact matching is substructure matching. As the name suggests, often chemists wish to query databases for molecules that contain a specific substructure. A substructure is a collection of atoms and bonds which are present in a molecule. Substructure searching is more complex than exact query matching due to the many possible orientations in which a substructure query may appear in a molecule. From graph theory, this problem is known as the subgraph isomorphism (SGI) problem. The SGI problem for chemical graphs means that a single substructure can be verified quickly to exists in the query molecule, however finding all substructures of a molecule to check scales quickly with an increasing number of atoms.

Initially, substructure searching relied on a simple backtracking computation on every compound in the database (Ray & Kirsch, 1957). The simple approach, however, is computational expensive and therefore becomes infeasible as the number of molecules in the database increases. Thus, as the number of molecules increased in chemical databases and the complexity of queries by chemists increased, it became clear that alternative approaches needed to be taken to allow substructure queries to occur. The two main approaches to improving the performance of substructure searching in chemical databases came through the usage of more performant subgraph isomorphism algorithms (Ullmann, 1976) and the use of screening (Adamson et al., 1973; Crowe et al., 1970; Feldman & Hodes, 1975; Hodes, 1976).

More advanced subgraph isomorphism exploits the chemical graph representation of molecules, an example of the Ullman algorithm (Ullmann, 1976). The Ullman algorithm relies on graph theory to convert molecules to adjacency matrices. An adjacency matrix of a chemical graph describes the connectivity between atoms and bonds. The adjacent matrices are then searched using a complex matching process. Ultimately, the implementation of Ullman's algorithm substantially improved the performance of substructure searching and was more effective than simple backtracking.

The second approach to improving performance in the SGI problem relied on the heuristic removal of molecules based on structural screening keys. A structural key is a feature present in a molecule and several structural keys make up a vector. The vector of structural keys is often referred to as a molecular fingerprint. Often molecular fingerprints are comprised of zeros and ones and are called bit vectors. Each molecule in the database is converted to its bit vector. The improved computation of substructure matching was achieved by converting the query substructure into a bit vector of structural keys and then comparing the query bit vector to the bit vectors stored in the chemical database. The idea of coarse filtering substantially reduced the number of compounds required to be searched using an SGI algorithm (Hodes, 1976).

2.2.2 Molecular fingerprints

Molecule are represented by descriptors for many chemoinformatics applications. A descriptor is a numerical representation of a molecule (Todeschini & Consonni, 2000). Descriptors can be intuitive chemical descriptions such as molecular weight (MW) or the number of hydrogen bond acceptors (HBA) or more esoteric descriptions such as structural fragments based upon analysis of the molecule or even quantum mechanical properties.

Due to their simple formulation and ease of use, molecular fingerprints have found application in many different chemoinformatics domains, such as similarity searching, virtual screening, QSAR, and scaffold hopping (Cereto-Massagué et al., 2015; Lewis & Wood, 2014; Vogt et al., 2010; Willett, 2011a). As a result, many different fingerprints encoding different information have been designed and created (Cereto-Massagué et al., 2015). Several key distinctions, however, separate the different types of the fingerprint. First, fingerprints can be separated into binary (bit), count, or continuous fingerprints depending on their encoding of chemical data. Binary fingerprints, as described above, encode the presence or absence of structural features. Count fingerprints encode the number of times a structural feature is observed. Finally, continuous fingerprints are based on continuous features of the molecule such as their physicochemical properties.

Further categorisation of fingerprints can be made according to the process of generating the molecular descriptors from which the fingerprint is composed. Structural keys, as described above, encode individual structural features, for examples the 166-bit Molecular ACCess System (MACCS) keys (Durant et al., 2002) and the PubChem fingerprint (*PubChem Substructure Fingerprint v1.3.*, 2018). A further category is a path or topology fingerprint. A path or topology fingerprint traverses the chemical graph and encodes paths of atoms and bonds up to a fixed length. The traversed paths are hashed and folded into a fingerprint of fixed length. Several approaches to topology-based fingerprints exist. One type of topological fingerprint is the Atom Pair (AP) fingerprint (Carhart et al., 1985). AP fingerprints capture the atom typing of a starting atom and then traverse the chemical graph to an end atom with an atom type. The number of bonds crossed between the starting and end atoms is also encoded (topological distance). Circular fingerprints are a further example of topological fingerprints but are different to standard path fingerprints due to relying on a circle of set radius, iteratively expanding to capture neighbourhood information. Notably, after capturing the neighbourhood information, the captured information is hashed and folded into a fixed-length fingerprint.

A significant hashed circular molecular fingerprint used in this work is the extended connectivity fingerprint (ECFP) (Rogers & Hahn, 2010). To calculate an ECFP, a process similar to the Morgan

algorithm for canonicalisation is employed. It is a non-trivial process; however, it can be summarised as follows. A 32-bit integer identifier is assigned to each atom. The identifiers are hashed using a hashing function. Next, a circle of radius one is drawn around each atom which captures the neighbours of the atom. The identifiers of all neighbours are then summed and hashed using the same hashing function. Again, another circle is drawn, this time of radius two. This captures more neighbours further away from each atom. This process of expanding the number of neighbours continues until a set value of radius is reached. The hashed identifiers are finally folded into a vector. Folding means applying modular arithmetic to the hashed identifier such that only the remainder is stored. The process of generating a Morgan fingerprint will return a molecular fingerprint whose bits are set based upon the substructural features of the molecule. An open-source implementation of the ECFP is used in this work, known as the Morgan fingerprint (*RDKit*, 2021).

2.2.3 Similarity search

An essential tenant of drug discovery is the concept that similar chemical structures maintain similar chemical properties. The tenant is often quoted as the "Similar Property Principle" (SPP) (Johnson & Maggiora, 1990). The SPP is an empirical observation that structurally similar molecules have similar bioactivity profiles. As with any empirical observation, this is not always the case (Lajiness, 1991). However, the SPP frequently motivates chemists to search through databases for similar compounds for various tasks (Willett et al., 1998). Performing a similarity search is similar to the process of substructure matching. First, a chemist has a query molecule and a list of molecules they wish to search through. Next, the query and list of molecules are converted to molecular fingerprints. Finally, the query molecule fingerprint is compared to each molecular fingerprint in the list. The comparison involves a similarity or distance computation between the query fingerprint and the molecular fingerprint in the list. Generally, similarity and distance computations can be divided based on whether the fingerprint is binary or continuous.

2.2.3.1 Metrics for binary descriptors

Similarity or distance metrics for binary descriptors are based on the counts of the number of bits set on ('1'), and, sometimes, also bits set off (0) in the query fingerprint and the reference fingerprint to be compared. For example, a common similarity and corresponding distance metric is the Tanimoto similarity coefficient (Tc) and the Soergel distance metric (Willett et al., 1998). The equations for the Tanimoto and Soergel distance metrics are given equation (1) below:

$$Tanimoto\ (Tc) = \frac{C}{A + B - C} \qquad Soergel = 1 - Tc \tag{1}$$

Where A is the number of bits set in binary fingerprint A, B is the number of bits set in binary fingerprint B, and C is the number of bits set in both binary fingerprints A and B. Thus, the Tanimoto value is bounded between zero and one. The Soergel distance is (1-Tanimoto) and is also bounded between zero and one.

The Tc has become a popular similarity measure due to good empirical performance on a wide range of tasks (Maggiora et al., 2013). In particular, Tc has become popular as an objective score for testing molecule design algorithms (Brown et al., 2019). While the Tc is a common approach for similarity searching in chemoinformatics, Tanimoto does come with some distinct limitations. Tanimoto tends to be biased towards the size of the molecule (Holliday et al., 2003). For example, if a molecule has many substructures that are hashed to unique bits in the fingerprint, the coefficient will become biased by the on bits compared to molecules of smaller size with fewer on bits. Additionally, at the Tanimoto distribution's tails, the molecules appear very dissimilar at low coefficient values or become almost equivalent structures at high coefficient values (Flower, 1998). In the case of using the Tc with the ECFP fingerprint, a low value of Tc can imply that molecules are structurally dissimilar, however, structures could maintain similarity in other representations. Lastly, distributions of Tanimoto tend to a mean value of 0.3, as reported by Godden et al. (Godden et al., 2000). This value was computed, utilising both simulated data of a fingerprint of 67 bits in length and experimentally utilising a pharmacophoric binary fingerprint of 1024 bits in length. Additionally, Todeschini et al. (Todeschini et al., 2012) demonstrated on simulated data of a binary fingerprint of 1024 bits in length, that the Tc value will have a mean of 0.4. However, this value will vary depending on the dataset under study and the number of bits set in the fingerprint.

It is essential to note several benefits of using the Tanimoto similarity, even with the limitations described. Firstly the calculation of the coefficient is fast due to not requiring any complex operations and therefore can be optimised using bitwise operations(Haque et al., 2011). Second, the Tc has been used successfully for the clustering molecules(Barnard & Downs, 1992; Butina, 1999) ; therefore, the Tc can be used as a nearest neighbour approximator similar to a QSAR model. As a result, for this work, the Tc will be leveraged throughout.

2.2.3.2 Continuous distance metrics

Often molecular descriptors are based on continuous descriptors. Similarity or distance computations are slightly more complex when using continuous-valued molecular fingerprints. One of the most popular distance measurements used in similarity searching is the Euclidean distance (Willett et al., 1998). The Euclidean distance is based on the triangular distance between two points. The merits of using the Euclidean distance is that the computation of Euclidean distance is relatively

easy to calculate, satisfies the triangle inequality, and relies on the low values present in the vector for comparison(Willett et al., 1998). A limitation of the Euclidean distance is that there is no upper limit on the distance that can be computed and thus it often needs to be normalised through further processing. Hence, when handling continuous-valued fingerprints, an alternative distance measure that can be used is the cosine distance (Willett et al., 1998). The cosine distance is based on the angle between two molecular fingerprints. The main benefit is that the cosine distance is bounded between [0,2]. However, the core limitation of the cosine distance is that it does not adhere to the triangle inequality, as zero dimensions are ignored (Willett et al., 1998). The equations for the Euclidean and cosine distances are given in equation (2):

$$Euclidean_{distance} = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2}, \qquad Cosine_{distance} = 1 - \frac{X \cdot Y}{\sqrt{X \cdot X}\sqrt{Y \cdot Y}} \quad (2)$$

Where X and Y are vectors, and X_i and Y_i are the ith element in the corresponding vectors.

2.3 Reaction representation

Chemical reactions are the physical interaction between two molecules. Chemists often wish to query collections of reactions and retrieve reaction information (Warr, 2014). In contrast to molecules, chemical reactions are substantially more complicated. Chemical reactions are composed of two parts—a set of reagents and a set of products. The set of reagents encompasses all the molecules which are reacted together. The set of reagents can include molecules known as agents needed for the reaction to occur but are not participatory such as catalysts or solvents. Finally, the set of products denotes all molecules generated during the chemical transformation, including the significant product and side products, such as gas or undesirable molecules. A reaction is read left to the right, with reagents on the left and products on the right. The separating arrow is called a reaction arrow. A depiction of a chemical reaction is shown in Table 2-1.

2.3.1 Reaction notations

The most popular approach for storing chemical reactions is the reaction SMILES (Weininger, 1988). A SMILES string represents each molecule in the reaction. Each molecule in reaction SMILES is separated by a '.'. The reagents are shown on the left. The reaction arrow showing the direction of the reaction is shown from reagents to products and is represented by a ">". The reaction agents are inserted in the middle to signify they will not be used during the reaction. Finally, the products are shown on the right. Table 2-1 below shows an example of a reaction SMILES for a bromination reaction (electrophilic aromatic substitution).

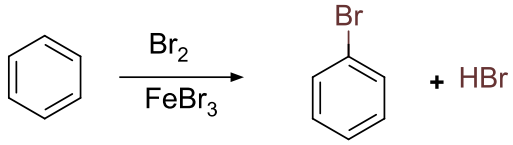
Reaction Depiction	
Reaction SMILES	C1=CC=CC=C1>BrBr.Br[Fe](Br)Br>BrC1=CC=CC=C1.Br
Reagents	C1=CC=CC=C1
Agents	BrBr.Br[Fe](Br)Br
Products	BrC1=CC=CC=C1.Br

Table 2-1 Shows a bromination reaction via electrophilic aromatic substitution. The reaction depiction is shown. The reaction SMILES (unmapped) for the complete reaction is shown. The reagents, agents, and products of the reaction in reaction SMILES are also shown.

An alternative linear approach to reaction representation is the Reaction International Chemical identifier (RInChI) (Grethe et al., 2018). The purpose of RInChI is the same as the InChI but extended to chemical reactions. RInChI aims to encode a unique representation for a given reaction. The RInChI follows the same hierarchical structure as InChI and uses software developed by the InChI Trust.

2.4 Reaction mapping

Often when storing and retrieving chemical reactions, the essential part of the transformation is the reaction centre. The reaction centre is the part of the reaction where bonds are being made and broken, and atoms change (Chen et al., 2013; Vléduts, 1963). Thus, defining a reaction centre is crucial as it facilitates the searching of chemical transformations rather than searching based on functionality in the starting material (SM) (the primary reagent) or the product (Warr, 2014).

A process known as atom-atom mapping (AAM or reaction mapping) identifies the reaction centre. AAM is a process where each atom in the reagents is given a number, and each atom in the products is given a number. The numbering is performed such that the atoms identified in reagents are tracked and given the same number in the products. An example of AAM is given in Table 2-2. The mapped reaction is an amide condensation. Atoms four and six are exchanged during the reaction.

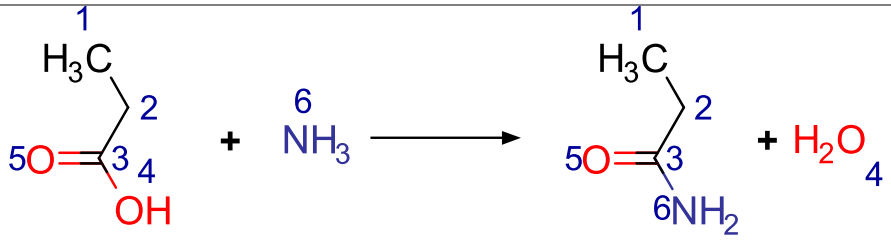
Reaction depiction with corresponding AAM.	
Reaction SMILES with AAM	[CH3:1][CH2:2][C:3]([OH:4])=[O:5].[NH3:6]>>[CH3:1][CH2:2][C:3]([NH2:6])=[O:5].[OH2:4]

Table 2-2 A depiction of an amide condensation with AAM is shown. The corresponding reaction SMILES with AAM is also shown.

AAM mapping has been explored extensively due to the necessity of identifying the reaction centre for various applications (Chen et al., 2013). The primary goal of AAM is that the mapping occurs without any human intervention and is therefore performed entirely using reaction mapping algorithms. Broadly reaction mapping can be split into three categories: structure, search, and heuristic-based AAM. Structure-based AAM uses processes derived from maximum common substructure searching to identify changing bonds and atoms (Apostolakis et al., 2008; Funatsu et al., 1988; Kumar & Maranas, 2014; Mcgregor & Willett, 1981; Rahman et al., 2016). In contrast, search-based AAM attempts to traverse a bond-making and breaking search space to find the shortest path between reactant and product (Akutsu, 2004; Litsa et al., 2019). Finally, heuristic-based methods rely on chemical heuristics to determine the most appropriate AAM (Jaworski et al., 2019).

2.5 Reaction databases

Similar to molecules, large chemical reaction databases have been compiled (Warr, 2014). Databases of chemical reactions can be split first into two categories: comprehensive and utility. Comprehensive databases cover an entire area of chemical reactions, such as those from a specific scientific journal (for example, CASREACT (Blake & Dana, 1990)). Utility databases are reaction databases dedicated to collecting reactions for a specific function, and therefore, the reactions are sourced from various locations without a guarantee of complete coverage (for example, Organic Syntheses (*Organic Syntheses*, 2021)). A further subdivision is a categorisation based on whether the databases are commercial or free. Commercial databases contain reactions that are pre-processed or stored in a machine-readable format (Reaxys (*Elsevier*, 2021)). Free-use databases are often sourced from publicly available stores of reactions such as patent literature (Lowe, 2012). Table 2-3 below describes several key reaction databases.

Database	**Availability**	**Source**	**Total number of reactions**
SPRESI (*InfoChem - DeepMatter*, 2021)	Commercial	Journals and Patens	>4.6M
Reaxsys (*Elsevier*, 2021)	Commercial	Journals, patents and Cross-indexed sources	>42M
CASREACT (*CAS Reactions*, 2021)	Commercial	Journals, patents, theses and Cross-indexed sources	>139M
Organic Syntheses (*Organic Syntheses*, 2021)	Free	Verified reactions	>5.5k
Open reaction database (*Open Reaction Database*, 2021)	Free	Patents, Submissions	>2.2M
WebReactions (*Openmolecules*, 2021)	Free	Merged databases	4.5M

Table 2-3 A table of free and commercial reaction databases. The databases are characterised by their source and total number of reactions.

2.6 Reaction classification

Chemists intuitively think about reactions based on different reaction classes. A reaction class is a label applied to a set of chemical reactions which maintain similar features. Therefore, reactions must be annotated so that chemists can quickly interrogate reaction databases and retrieve known transformations. Unlike molecules that have either an IUPAC name or an InChI, reactions have no set labelling system. Several labelling systems have been developed, such as RXNO, which the Royal Society of Chemistry created to standardise the reaction nomenclature (*Rsc-Ontologies/Rxno: Name Reaction Ontology*, n.d.). Recently, the Sheffield Hierarchical REaction Classification (SHREC) (Ghiandoni et al., 2019) has been proposed whereby each reaction is labelled using a hierarchical label which can be broken down into separate classes. With the large numbers of reactions being generated by chemists at an ever-increasing rate, the goal of reaction classification is to automate classifying reactions without the need for human labelling. Different approaches to reaction classification have been explored extensively. The main categories are model-driven approaches relying on expert rules or representations to separate reactions into different classes, and data-driven approaches relying on an analysis of reaction data to separate the reactions into classes. Early attempts relied on tracking bond-making and breaking during reactions which were then annotated with symbols based on expert rules (Theilheimer, 1960). Further work extended the model approach for specific cyclic reactions (Balaban, 1967). This approach for cyclic reactions was then later generalised to include more complex pericyclic reactions (Hendrickson, 1997). By annotating reactions with bond breaking and making information, reactions could be sorted into different categories. Similarly, an approach to storing reactions as hypergraphs was developed (Vléduts, 1963). Hypergraphs are the fusion of multiple graphs into one large, connected structure. This connected structure contains the reaction centre and surrounding environment related to the reaction. Storing reactions as hypergraphs facilitated easier extraction of reaction features, which could then be used to group reactions together in databases. Further extensions to the concept of reaction hypergraphs were explored by (Fujita, 1986) and then later (Hendrickson, 1997). Finally, in terms of modern model-based approaches to reaction classification, the concept of the reaction hypergraph was reintroduced to the literature as part of the condensed graph of reaction (CGR) (De Luca et al., 2012; Nugmanov et al., 2019; Varnek et al., 2005).

In contrast to the reaction hypergraph approaches, a different popular approach to model-driven reaction classification is the Dugundji-Ugi model (Ugi & Dugundji, 1973). The model is based on the creation of a bond and electron matrix to describe the molecules. The goal of the Dugundji-Ugi model is to encode the reagents, reaction transformation, and end products in a series of matrices

(B, R, E). The reagent matrix (beginning) encodes the bonds between the atoms of the SM, the reaction transformation matrix captures the change in the bonds based on bond order, and the product matrix encodes the bonds between the atoms of the products. The Dugundji-Ugi model forms part of several reaction searching systems (Gasteiger et al., 1987, 2002).

A limitation of the model-based approach is that the models are limited to chemistry relating directly to the reaction centre. If reactions are influenced by other parts of the molecules not in the reaction centre, then model driven approaches do not capture that information well. Hence, rather than enforcing a reaction model, it is instead more flexible to allow for transformations to naturally group dependent upon on reaction data directly.

Data driven approaches, rely on the reaction data itself to separate out distinct classes of transformation. Moreover, as the size and complexity of reaction data has increased data driven approaches have become more popular (Warr, 2014). Data-driven approaches to reaction classification use the concept of knowledge discovery where features are derived from reaction examples directly (Chen et al., 2013).

The first approaches to data-driven reaction classification relied on creating reaction graph data structures directly from reaction example based on connected graphs of reagents and products. These connected graph data structures where then mined for generalised features. By utilising the reaction examples directly, this allows for a natural grouping of reactions (Wilcox & Levinson, 1986). The approach was further extended as part of the SYNCHEM (Gelernter et al., 2002) and RETROSYN (Blurock, 1990) knowledge bases.

One of the first approaches to classification of reactions via molecular descriptors was HORACE (Rose & Gasteiger, 1994). HORACE's ability to classify reactions was further improved by the inclusion of more advanced physical-chemical features. The introduction of self-organising maps (Chen & Gasteiger, 1997) further enhanced the approach by allowing clustering and better visualisation of the reaction space. Self-organising maps (SOMs) rely on networks of functions to create clusters of reactions relying on the physical-chemical features of reactions.

Alternatively, rather than relying on physical-chemical descriptors, topological or path descriptors can be used instead. For example, CLASSIFY (Kraut et al., 2013) by infoChem utilises hashing of the reaction centre based on spheres of increasing size. CLASSIFY builds the hash codes based on different sized spheres surrounding atoms in the reaction centre in an automated fashion depending on how much information is required. The unique hash codes can then be identified and grouped.

This approach has been further extended as part of the Reaxys database. However, a fundamental limitation of the CLASSIFY approach is that stereochemistry is lost during the hashing process.

An alternative approach to encoding topological information of a reaction is a difference vector (Daylight Chemical Information Systems Inc, n.d.) based upon atom pairs encoded into a data structure. Reaction vectors (RVs) are a form of difference vector that rely on the difference between starting material (SM) atom pairs and product atom pairs. RVs were patented by (Broughton et al., 2003) and later adapted and improved for molecular structure generation, reaction analysis, and classification (Ghiandoni, 2019; Ghiandoni et al., 2019, 2020; Hristozov et al., 2011; Patel, 2009; Patel et al., 2009; Wallace, 2016). Due to their importance as a method of structure generation that is central to in the work of this thesis, a comprehensive description of RVs and their development is given in Chapter 4. Finally, in related work, Schneider et al also explored alternative approaches to encoding topological difference vectors and developed a corresponding machine learning model to classify reactions based upon fingerprints (Schneider et al., 2015).

As reaction datasets have become more extensive and more readily available (D. M. Lowe, 2012), deep learning has become more popular to aid the construction of reaction classification models. For example, recent work uses deep neural networks relying on reaction SMILES as the input to the network to classify reactions on a cleaned public patent reaction set (Schwaller et al., 2021). Furthermore, additional applications of deep neural networks as part of a reaction classification scheme have been proposed, such as leveraging deep highway networks trained on products from retrosynthetic analysis (Baylon et al., 2019).

2.7 Synthesis planning

An essential motivation for creating large reaction databases has been facilitating and aiding chemists in designing synthetic routes to molecules. A synthetic route is a sequential path of reactions that a chemist carries out to synthesise a desired compound. A synthetic route can be single-step or multi-step. The process of designing a synthetic route on a computer is known as computer-aided synthesis planning (CASP).

One of the most common approaches to synthesis planning is retrosynthesis. Retrosynthesis starts from the desired molecule that needs to be made. Disconnects on the molecule are then made. This forms a tree data structure where the nodes consist of molecules and the edges represent disconnections. Importantly, disconnections should be reversible such that the disconnected molecule can be recombined using a forward reaction. The tree is grown until a molecule is generated that cannot be disconnected or is available in a collection of SMs; this is a leaf node. The path from the leaf node to the root node is then traversed, forming the synthetic scheme. E. J. Corey

(Corey, 1967) first described the concept of retrosynthesis. However, the first tool to use retrosynthesis was the LHASA program (Corey et al., 1972). LHASA used hand encoded reactions to assist chemists in the process of retrosynthesis. Possible disconnections were suggested to the chemist based upon the functionality present in the molecule. LHASA was improved over time by the inclusion of a more extensive knowledge base of disconnections. As the knowledge base grew, there was a combinatorial explosion in the number of possible disconnections and, therefore, several heuristics was also included to prioritise specific disconnections. However, these heuristics had to be hand encoded. An important limitation of the LHASA program was that a trained chemist needed to interact with the program to navigate the tree of disconnections to form a path from starting material to product. As a result, alternative programs such as SYNCHEM (Gelernter et al., 1977) and later SYNCHEM2 (Agarwal et al., 1978) were developed. SYNCHEM used hand encoded reaction libraries and faster searching of the retrosynthetic tree utilising better heuristics coupled with parallel computing. Importantly, however, SYNCHEM was autonomous and did not rely on human interaction.

Both LHASA and SYNCHEM utilised a knowledge base of reaction transformations. In addition, approaches utilising the Dugundji-Ugi model were developed which aimed to discover novel reactions to propose synthetic pathways, without the need for explicit knowledge bases, for example, IGOR (Ugi et al., 1993) and EROS (Gasteiger et al., 1987).

Recently, machine learning has become a popular approach to synthesis planning. A notable example is AlphaChem (Segler et al., 2018) which relies on deep neural networks to prioritise reaction templates used to disconnect molecules. AlphaChem coupled with a Monte Carlo Tree Search (MCTS) approach to prioritise possible reactions has been demonstrated as able to propose synthetic schemes to molecules of similar quality as PhD level chemists. The MCTS approach is further described in Chapter 6. Furthermore, the concept of utilising neural networks have been proposed for retrosynthetic planning relying on learnt rewards (Schreck et al., 2019). Finally, an alternative approach to synthetic planning is part of the Chematica program. Chematica (Szymkuć et al., 2016) utilises a large corpus of synthetic transformations (~50k) hand encoded by a team of expert chemists. Moreover, this extensive knowledge base has been coupled with large databases of reactions to form an interconnected network of organic chemistry. Potential synthetic routes to molecules are found by traversing paths in the network.

It should be noted that while synthesis planning is an essential tool in the synthetic chemist's arsenal, it cannot be used to design molecules from scratch based on a pre-defined property profile. However, synthesis planning tools can be coupled with de novo design whereby a molecule is first

designed from scratch (de novo design) and then a synthetic planning tool is used to propose synthesis routes to the designed compound.

2.8 Conclusions

Molecule and chemical reaction representation is a core part of chemical information systems. This chapter first described different molecular representations starting with linear and non-linear representations and then detailing canonicalisation approaches and strategies to convert molecules into fingerprint descriptors. The second part of this chapter focussed on the reaction representation and overviewed several approaches to encoding reaction information. Finally, a review of synthesis planning and different approaches to retrosynthesis were given. The next chapter explores the idea of de novo design and the expansive ecosystem of software available to explore and search chemical space for new molecules.

Chapter 3 De novo design

De novo design is the sub-discipline of chemoinformatics focussed on molecular design. The goal of de novo design is to construct novel molecules which fit a predefined property profile. A predefined property profile is the biological and physical-chemical characteristics that the designed molecule must maintain. To generate molecules of interest, a de novo design software traverses the chemical space which is the search space that contains all possible molecules. Chemical space is vast. Drug-like chemical space, a subset of chemical space that contains all possible drug molecules, and thus only a fraction of all possible molecules, has been estimated to be 10^{60} molecules (Bohacek et al., 1996), a veritable universe of small molecules. Hence, de novo design software must be carefully constructed to identify molecules that fulfil a predefined property profile effectively. Principally de novo design software comprises three parts: a molecule scoring module; a molecule construction module; and a search and optimisation module. Molecule scoring aims to assess molecules based on objectives that are part of the property profile. Molecule construction is the process of how the molecules are built in-silico. Finally, search and optimisation aim to combine scoring and construction to identify a path to regions of interest. Each module must be carefully designed to identify desired molecules in the chemical space.

This chapter focusses on the wide variety of different approaches to de novo design that have been developed in recent decades (Schneider & Fechner, 2005) and is separated into five parts. First, molecule scoring approaches are reviewed exploring single parameter and then multi-parameter optimisation, and then ligand- and receptor-based scoring. Second, an overview of molecule construction is given, including atom, fragment, and reaction-based construction methods. Third, search and optimisation is then reviewed, including stochastic, evolutionary and then deterministic methods. Fourth, generative design methods for de novo design are reviewed, including deep neural network-based approaches. Finally, validation and testing of de novo design approaches are discussed.

3.1 Molecular scoring

Molecule scoring is the process of converting a molecule into a number or vector of numbers that is then compared to the design objective. The design objective is often a pre-defined property profile which is the ideal values of the properties which the designed compounds should have. A de novo design software will use an objective scoring function based on the property profile. Broadly, scoring functions are split into two categories. Single objective problems (SOP) and multiobjective problems (MOP) (Liu et al., 2021; Nicolaou & Brown, 2013). MOPs are more complex due to the competition between objectives. In a drug design campaign, a fusion of different scoring methods are required. Multiobjective approaches attempt to score a molecule either via the merging of individual objectives in a process known as scalarisation (Collette & Siarry, 2004), or via the assessment of a pareto front (Gillet et al., 2002). Scalarisation relies on the use of a function to combine multiple objectives together. A simple approach to scalarisation is to take the average of the objectives, however, this can lead to a single objective compensating for harder to optimise objectives. An alternate approach is to utilise the geometric mean. The geometric mean is part of several scoring functions in the GuacaMol de novo design benching marking suite (Brown et al., 2019).The geometric mean is particularly useful for scalarisation due to any objectives with a value of zero instantly leading to a molecule scoring zero overall and it, therefore, requires that all objectives are satisfied to some extent. However, frequently more complex approaches are used, for example, a weighted summation of objectives (Nicolaou et al., 2007), although how the relative weights of the objectives are determined is non-trivial.

A pareto front assessment is based on assessment of non-dominated solutions. Having a non-dominated solution means that any improvement to any objective is to the detriment of the other objectives. All solutions that exist behind the pareto front are considered “dominated”. However, as more and more objectives are added it is harder to obtain balanced solutions. Finally, for non-trivial problems, there is also a computational performance cost when using such methods.

In practice, whether the goal is scoring via SOP or MOP, the scoring objectives can be divided into two categories; ligand-based scoring and receptor-based scoring (Hartenfeller & Schneider, 2011a). Ligand-based scoring often requires the usage of chemical descriptors or similarity searching to form the objectives. In receptor-based scoring, the objective value is derived from the docking of a ligand to a target protein. The aim of docking is to virtually position a molecule into a target protein in an attempt to mimic the binding of the molecule in a real protein (Meng et al., 2012). Receptor-based scoring relies on different metrics used to assess fit of the molecule in the target. Receptor-based objectives have a more significant computational cost compared to ligand-based objectives.

In the de novo design ecosystem, a wide variety of different software have been developed using different scoring approaches. The first examples of de novo design software utilised receptor-based scoring such as HSite/2D skeletons (Danziger & Dean, 1989a; Lewis & Dean, 1989a, 1989b) and 3D Skeletons (Gillet et al., 1990). Remarkably, even with a more considerable computational cost the trend of utilising receptor-based approaches to scoring continued with LUDI (Böhm, 1992a, 1992b, 1993, 1994), NEWLEAD (Tschinke & Cohen, 1993), SPLICE (Ho & Marshall, 1993),GroupBuild (Rotstein & Murcko, 1993b), CONCEPTS (Pearlman & Murcko, 1993) and SPROUT (Gillet et al., 1993, 1994; Mata et al., 1995) The first published approach of using a ligand-based approach was in "Chemical Genesis" (Glen & Payne, 1995). In more recent work, objectives in de novo design have primarily focussed on ligand-based scoring due to the methods generating large numbers of compounds (Bradshaw et al., 2020; Guimaraes et al., 2017; Olivecrona et al., 2017; Yang et al., 2017).

Further examples of ligand- and receptor-based scoring in de novo design are given below.

3.2 Ligand-Based methods

Ligand-based scoring objectives exploit the SPP via molecular descriptors, and involve comparing generated compounds to a reference compound (Hartenfeller & Schneider, 2011b). The calculated molecular descriptors could,for example, be atomic mass weight (AMW) or partition coefficient (LogP). Alternatively, ligand-based objective functions could also utilise 2D similarity metrics derived from molecular fingerprints derived from molecular fingerprints. Several de novo design softwares have relied on the use of ligand-based objective scoring such as Chemical Genesis (Glen & Payne, 1995), SkelGen (Dean et al., 2006; Stahl et al., 2002; Todorov & Dean, 1997), Nachbar (Nachbar, 1998, 2000), FLUX (Fechner & Schneider, 2006, 2007), DOGS (Hartenfeller et al., 2012), Molecule Evoluator (Lameijer et al., 2006), CReM (Polishchuk, 2020), and DINGOS (Button et al., 2019). Furthermore, 3D-based methods (Damewood et al., 2010; Proschak et al., 2009) reliant on the 3D structure of the reference compound have been proposed.

Ligand-based methods are generally the most computationally efficient objectives used in de novo design as they rely on methods such as similarity searching, substructure matching or descriptor calculations which are highly optimised. Additionally as the amount of data has increased overtime quantitative structure activity relationship (QSAR) models have become popular as part of ligand-based scoring to drive the search towards regions of chemical space that are promising (Besnard et al., 2012).QSAR models rely on building a function which maps between structure and biological activity. These models are often built from molecular fingerprints which are fast to compute.

Recent work has also suggested that over reliance on the SPP using ligand based methods prevents de novo design methods from accessing more novel chemical space (Thomas et al., 2021).

3.3 Receptor-Based Methods

Receptor-based scoring attempt to assess a molecule's fit to a particular binding site. The approaches of fitting a molecule to a binding in site and are similar to virtual screening docking approaches. In principle, there are three main strategies for receptor-based molecular scoring that can be used: molecular force fields; empirical functions; and knowledge-based scoring functions (Hartenfeller & Schneider, 2011b). Remarkably, early de novo design software such as HSITE (Danziger & Dean, 1989b), CONCEPTS (Pearlman & Murcko, 1993), and SPROUT (Gillet et al., 1993, 1994; Mata et al., 1995) all relied on the use of receptor-based strategies. Due to the ubiquitous availability of bioactivity data and the rise of generative methods (See 3.6) these approaches have fallen out of popularity. On the other hand, as highlighted earlier, recent work (Thomas et al., 2021) suggests that using receptor-based scoring functions may lead to accessing more novel chemical space. It is therefore instructive to illustrate the different approaches to receptor-based scoring.

3.3.1 Molecular force fields

In de novo design (and virtual screening) there is a requirement to obtain scores quickly with a user-defined degree of accuracy. In de novo design due to the large number of molecules being generated, molecular force fields are helpful due to the fast compute time and their customisability.

At the core of any molecular force field approach lies the exploitation of simple physcial properties. In particular, force fields rely on physical principles, to model intra- and intermolecular interactions and are based on a "Ball and Spring" model to represent the atoms and bonds. Here the atoms are treated as spheres with differing degrees of hardness and the bonds are treated as springs with different degrees of stiffness. Within this setup, the energy of the molecule is obtained through the summation of the individual contributing energy terms.

An example of a generalised force field model is shown below.

$$E_{FF} = E_{str} + E_{bend} + E_{tors} + E_{vdw} + E_{el} + E_{cross} \quad (3)$$

Where:

$$E_{str} = Energy\ from\ stretching\ a\ bond$$
$$E_{bend} = Energy\ from\ bending\ a\ bond$$
$$E_{tors} = Torsional\ energy$$
$$E_{vdw}, E_{el} = Enegy\ of\ Van\ der\ Waals\ interactions, Energy\ of\ Electrostatic\ interaction$$
$$E_{cross} = coupling\ between\ the\ stretching, bending\ and\ torsional\ compoenents$$

Different methods of calculating each term give rise different force fields. When a molecule is generated, molecular force field scoring aims to return the energy of a 3D pose of the molecule. Generating a 3D pose however, is non-trivial (Hawkins, 2017). Often multiple poses will be calculated and inserted into a binding site. Hence, a trade-off exists between fast scoring and time spent generating better poses. Force-field based approaches to de novo design were first explored with the LEGEND algorithm (Nishibata & Itai, 1991, 1993). Force field based approaches were further explored in a "fragment shuffling" algorithm which used a force field approach to estimate energies which were then added to the atoms' meta-data to decide on fragments to proritise (Nisius & Rester, 2009).

3.3.1.1 Empirical Scoring

Empirical scoring attempts to derive an equation that relates interaction terms to the binding energies based on a set of protein-ligand complexes. Empirical scoring can be as simple as counting the number of hydrogen bond donors and hydrogen bond acceptors of receptor and ligand. More complex multi-component methods may involve the surface area interactions between the ligand and receptor. The advantage of using an empirical scoring function over a force-field is that it can be faster to compute. A significant limitation of the empirical scoring function approach is that the scoring function is highly dependent upon the training data used to derive the function (Guedes et al., 2018). However, when care is taken in building an appropriate empirical scoring function, a substantial computation time save can be achieved. LUDI (Böhm, 1992a) first used empircal scoring functions as part of its objective scoring function. More recently, the LigBuilder software (Wang et al., 2000; Yuan et al., 2011, 2020) utilises an empirical scoring function.

3.3.1.2 Knowledge-based scoring functions

Knowledge-based scoring functions (Muegge, 2006) are based around statistical approximations of interactions with binding sites. A knowledge-based scoring function is computed by taking a database of ligand-protein complexes either from sources such as small molecule crystal structures or large curated databases, for example, the Cambridge structural database or the Protein Data Bank (PBD) and using them to parameterise the force field potentials. The principle of knowledge-based methods is that statistical potentials can be obtained via careful data mining and balancing of the ligand-receptor binding interactions. Like empirically derived receptor-based scoring functions, knowledge-based scoring functions require careful tuning and selection of appropriate examples to be able to achieve competitive performance. Several examples of software which use knowledge-based scoring functions are SMoG (DeWitte et al., 1997; DeWitte & Shakhnovich, 1996; Ishchenko & Shakhnovich, 2002), DrugScore (Gohlke et al., 2000), and Drug Score eXtended (DSX) (Neudert & Klebe, 2011)

3.3.1.3 *Pharmacophore methods*

An alternative approach to receptor-based scoring is the usage of pharmacophore modelling (Reddy et al., 2013; Wermuth et al., 1998). A pharmacophore is an arrangement of features in 3D space that conserves the essential binding structural features. The molecules are scored using their fit to the spatial arrangement of binding features. Several de novo design tools have been created to exploit pharmacophore features. Examples include SkelGen (Dean et al., 2006; Stahl et al., 2002; Todorov & Dean, 1997) and PhDD (Huang et al., 2010). In particular, PhDD, relies heavily on pharmacophoric modelling for positioning of fragments in a binding pocket. It is important to note that often pharmacophore scoring is also used as a ligand scoring approach, such as in DOGS (Hartenfeller et al., 2012) and DINGOS (Button et al., 2019), which used 2D pharmacophoric based descriptors.

3.4 Molecule construction

Molecule construction is the approach taken to build the molecule in-silico. Broadly, three categories of construction exist; atom based, fragment based, and reaction-based (Meyers et al., 2021). Atom-based methods construct molecules atom-by-atom. Fragment-based approaches bind together individual molecular fragments to build molecules. Reaction-based approaches use synthetic transformations to combine molecules together to form novel structures. Figure 3-1, shows a schematic of the different approaches to molecule construction. The first is atom-based (top) where a bromine is added to a six membered ring. Second is fragment based (middle), where a fragment is added to a six membered ring. Third, is reaction-based, where a bromine is added to an aromatic ring via a bromination reaction.

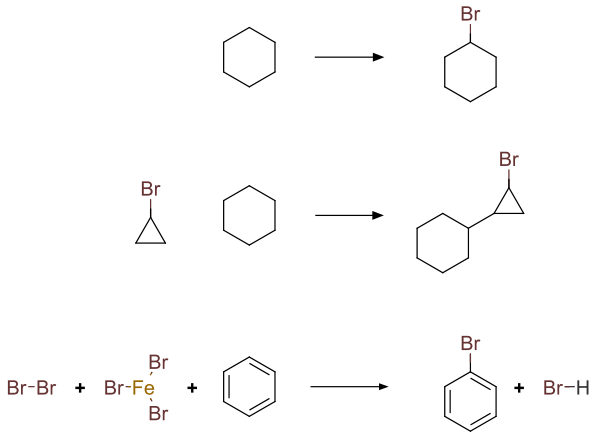


Figure 3-1 Schematic of different approaches to molecule construction. Atom based (top), fragment based (middle), and reaction-based bottom.

3.4.1 Atom-based methods

Atom-by-atom methods, as the name suggests, are a form of molecule construction where a molecule is designed by adding a single atom each time. Atom-by-atom methods, start from either an empty graph, a seed atom, or from a starting fragment and grow by adding an atom sequentially to form the molecule. The growth can aim to build an entirely new molecule or link two separate molecules together. Atom-by-atom molecule generation allows for fine-grained control of the molecule being constructed. Atom-by-atom methods were one of the first de novo design molecule construction approaches to be implemented, for example, the CONCEPTS method uses this approach (Pearlman & Murcko, 1993). Atom-based methods can rely on using the molecular graph representation as part of their construction method and several modern implementations of atom-based de novo design have used the graph-based representation such as Molpher (Hoksza et al., 2014) and GB-GA (Jensen, 2019).

In contrast to atom-based methods utilising molecular graphs, in 1993 a patent was filled and later granted on the basis of molecule construction using SMILES notations (Weininger, 1995). SMILES based-atom construction relies on the creation of the linear SMILES notation character-by-character. This was an attractive approach as it allowed a large number of structures to be generated with little computation expense and it has regained popularity with several recent works using SMILES as their construction medium (Kwon & Lee, 2021; Yang et al., 2017; Yoshikawa et al., 2018).

De novo design using atom based methods are often limited by the “Combinatorial Explosion” of possible configurations of atoms that could be constructed for a molecule of a reasonable size. For example, estimates of the size of drug-like chemical space vary from 10^{23} to 10^{100} molecules, with the common value being 10^{60} molecules (Bohacek et al., 1996; Ertl, 2003; Polishchuk et al., 2013). Hence, while atom-based methods afford faster computation they come at a considerable cost. Remarkably, however, atom-based methods have seen a resurgance in popularity due to the popularity of deep learning based approaches which are described in section 3.6.

3.4.2 Fragment-based methods

Fragment-based methods use parts of molecules as fragments to construct molecules. Fragments can range from complex ring systems to single atoms. Moreover, the concept of fragment-based de novo design, is firmly supported by the broader fragment-based drug discovery paradigm (Murray & Rees, 2009). Fragment-based methods have been used in molecule construction since the beginning of de novo design, with both HSITE (Danziger & Dean, 1989b) and 3D Skeletons (Gillet et al., 1990) using carbon-based fragments.

Fragment-based methods utilise large collections of fragments, either created by breaking apart (fragmentation) (Lewell et al., 1998; Liu et al., 2017) of prexisting molecules or the use of small molecules which exist in pre-existing databases (Irwin et al., 2020). The benefits of using these fragments is that they are derived directly from known molecules so that there is an increased liklihood (but no guarantee) that the final constructed molecule is chemically feasible (that is satisfies chemical valance rules) and synthetically tractable.

One of the essential benefits of fragment-based drug discovery is the flexibility associated with the pool of fragments. Modern fragment-based construction methods such as CReM (Polishchuk, 2020) utilise fragment libraries in the 10s of millions of distinct fragments. Moreover, fragment-based construction methods depend on the fragment pool that is utilised. As different molecules can be produced depending on which library is utilised this allows a focus to be placed on particular requirements such as synthetic accessibility (Ertl & Schuffenhauer, 2009) or commercial availability (Enamine, 2021).

Due to the attractive nature of fragment-based de novo design methods, an extensive ecosystem of fragment-based de novo design software has been developed including: LUDI (Böhm, 1993), CONCERTS (Pearlman & Murcko, 1996), SKELGEN (Dean et al., 2006; Stahl et al., 2002; Todorov & Dean, 1997), COLIBREE (Hartenfeller et al., 2008), GANDI (Dey & Caflisch, 2008), ADAPT (Pegg et al., 2001), FLUX (Fechner & Schneider, 2006, 2007), TOPAS (Schneider et al., 2000), LEA3D (Douguet et al., 2005), LigBuilder (Wang et al., 2000; Yuan et al., 2011, 2020), OpenGrowth (Chéron et al., 2015), MOARF (Firth et al., 2015) and LiGen (Beccari et al., 2013).

There are numerous different ways in which fragments can be combined to form valid molecules. These methods encompass three general classes; fragment growing, fragment linking, and fragment shuffling (Hartenfeller & Schneider, 2011b). Fragment growing aims to grow a single fragment by addition of new fragments. Fragment linking aims to link together two separate fragments such that the new fragment is a combination of the two designed ones. Additionally, more complex fragment combination strategies exist and will be described separately.

3.4.2.1 Growing

Fragment growing, as the name suggests, aims to grow fragments in a sequential manner. Fragment growing builds molecules up one fragment at a time, usually within a boundary which the molecule is confined to such as a binding pocket. Several examples of the growing strategy in fragment-based de novo design are Groupbuild (Rotstein & Murcko, 1993b), SMoG (DeWitte et al., 1997; DeWitte & Shakhnovich, 1996), FlexNovo (Degen & Rarey, 2006) and OpenGrowth (Chéron et al., 2015).

Fragment growing is essentially a coarser version of atom-by-atom construction, except in fragment growing the building blocks are larger. Therefore, fragment growing can obviously lead to molecules being generated which are chemically unfeasible as the individual fragments cannot coexist in the same molecule. Moreover, the bonds that connect the fragments together may not be possible and therefore, synthetically intractable. In addition to issues surrounding synthetic tractability a key difficulty is the ordering by which the fragment growing occurs. This ordering is often dictated by a search algorithm for which there are many and which are described in more detail in section (3.5).

3.4.2.2 Fragment Linking

An alternative to fragment growing is fragment linking. Fragment linking aims to join fragments together by using an intermediary fragment to link the fragments together. This intermediary fragment is called a linker. Often large databases of linkers will be used, and constraints based upon 3D geometry or 2D geometry are utilised. Fragment linking further constrains the search space by forcing a substructural requirement on the designed molecules, that is, the designed molecules must contain the desired starting fragments.

A difficulty of using fragment-based methods is that exploring the orientation of the individual building blocks is not trivial. One of the first applications of this technique was LUDI (Böhm, 1993), where a decision tree of hard-coded rules was used to generate interaction sites. Following this, the interaction sites were linked with small fragments which were then linked to create the molecules. A more modern approach to fragment linking was developed as part of GANDI (Dey & Caflisch, 2008) and AutoGrow (Durrant et al., 2009, 2013; Spiegel & Durrant, 2020), where fragment are linked using reaction-based de novo design (see section 3.4.3).

3.4.2.3 Advanced Methods of Fragment based construction

Over time, several more advanced approaches to fragment construction have been developed. These methods can be seen as derivatives of growing and linking. A notable method is the BREED algorithm (Pierce et al., 2004) which uses a superposition of two ligands and identifies key strategic bonds. The fragments attached to the key strategic bonds are then swapped to produce two new molecules. In a similar manner to BREED, fragment shuffling (Nisius & Rester, 2009) performs a similar process of identification of key fragments which are then interchanged. A significant limitation with fragment shuffling techniques, as with most fragment-based approaches, is that the superposition or swapping process of key fragments does not consider synthetic accessibility during construction. Therefore, unless synthetically accessibility is included within the scoring function in some way, there is no guarantee that the molecule is synthetically accessible in the lab.

An alternative method of growing is to grow across a grid of points for example in the BUILDER software (Lewis et al., 1992). BUILDER is part of the lattice sampling class of fragment-based construction methods. BUILDER took docked fragments and constructed an irregular series of lattice points based on the docked fragments. The irregular lattice of points were then sampled. The sampled lattice points were then overlaid with fragments to construct new molecules. In 1995 the second iteration of BUILDERv2 (Roe & Kuntz, 1995) was released. Several improvements to the BUILDER strategy were added. The first improvement was to the construction and sampling of the lattice. The second improvement was the introduction of an improved pathfinding algorithm to link individual docked fragments together. Further small updates included better atom typing and bond characterisation.

3.4.3 Reaction-Based de novo design

In the lab molecules are constructed using reactions. A reaction is a transformation a chemist will carry out that modifies one molecule to form another (see section 2.3). Reaction-based de novo design therefore uses reaction transformations to construct molecules in a logical discrete manner. Reaction-based de novo design is a beneficial approach to molecule construction as the molecules have a greater chance of being synthetically tractable. Moreover, by using reaction transformations it decreases the overall size of the search space to molecules which should have a higher degree of synthetic tractability compared to other construction approaches. As reaction collections increase in size, the size of the available space of molecules that can be constructed increases. The reaction transformations used in reaction-based de novo design can be as simple as adding an atom, or incredibly complex such as the formation of novel ring system. Reaction-based de novo design utilises templates for reactions. These templates can be derived from data or be handcrafted. Early examples, including TOPAS (Schneider et al., 2000), SYNOPSIS (Vinkers et al., 2003), and FLUX (Fechner & Schneider, 2006, 2007) relied on small (<20) highly curated reaction templates. Templates tend to focus on reaction centres and ignore the surrounding molecule (which could prevent the reaction occurring in the lab). In reaction vectors (RVs), a prominent approach for reaction-based structure generation (Ghiandoni, 2019; Gillet et al., 2013; Patel, 2009; Patel et al., 2009; Wallace, 2016), recent work has been developed to overcome these limitations via the use of machine learning (Ghiandoni et al., 2020). Finally, a series of reactions which are used to construct a new molecule should be transferable to the laboratory with minimal alteration, thus avoiding the need for synthetic planning tools entirely.

A number of de novo design methods have been developed that construct molecules by applying transformations based on known reactions such as, AutoGrow (Durrant et al., 2009, 2013; Spiegel & Durrant, 2020), DOGS (Hartenfeller et al., 2012), DINGOS (Button et al., 2019),Molecule Chef

(Bradshaw et al., 2019), ChemBO (Korovina et al., 2019), Apollo1060 (Gottipati et al., 2020), REACTOR (Horwood & Noutahi, 2020) and Synthesis DaGs (Bradshaw et al., 2020). Particularly important to this thesis are RVs. RVs encode transformations directly through the use of reaction collections rather than the hand crafted approaches to reaction selection such as that of DOGS, DINGOS and AutoGrow (Button et al., 2019; Hartenfeller et al., 2012). Moreover, the process of using RVs as part of a structure generation approach has developed substantially over recent years and a detailed overview of the approach is given in Chapter 4.

3.5 Searching and solution optimisation

The final part of a de novo design software is the search component. In essence, to build compounds with desired properties, a sequence of construction steps needs to be created and this leads to what is often referred to as the "combinatorial optimisation" problem. The sequence will transform the starting point (a blank graph, fragment, or molecule), over successive construction steps to a final designed compound. However, there are many possibilities to explore at each construction step with the number of distinct molecules being grown increasing exponentially at each step. Search strategies are therefore needed to traverse the possible chemical space accessible from the starting point to compounds which maximise the given property (Hartenfeller & Schneider, 2011b; Meyers et al., 2021). A major limitation in traversing chemical space, besides its vastness, is that chemical space is not smooth (Lajiness, 1991). This leads to chemical space containing many optima. As a result of the inherent roughness, it is therefore important that methods can move from local minima, which are merely regions in chemical space having properties which are desirable but may not be the best solution. Ultimately the goal when searching chemical spaces is to find the global minima. Consequently, it is imperative that searching methods can jump out of local minima and be able to identify the global minima correctly and many different strategies have been developed that aim to navigate the chemical space effectively. Search strategies can be split into stochastic and deterministic search.

3.5.1 Stochastic search methods

Stochastic search leverages the use of randomness to traverse the chemical space which allows a search to escape local optima by sampling probability distributions. To leverage stochastic search, the Metropolis algorithm can be employed (Metropolis et al., 1953). The Metropolis algorithm accepts changes to a molecule depending on the scoring criterion. Equation (4) for the Metropolis algorithm is given below.

$$P(accept) = \min\left(1, e^{-\Delta Score}\right) \quad (4)$$

In the context of de novo design P(accept) is the probability of accepting a change to the molecule. $\Delta Score$ is the change in score between starting material and product. A limitation of the Metropolis algorithm is that it is easily trapped in local optima at the start of the search due to relying on delta score to form the probability distribution. Hence, it is beneficial to slowly decrease the amount of exploration over time, thus allowing the search to explore widely and then slowly converge upon an optimum. This is performed by including a temperature T. This algorithm is known as simulated annealing and is shown in equation (5) below (Kirkpatrick et al., 1983).

$$P(accept) = \min\left(1, e^{-\frac{\Delta score}{T}}\right) \quad (5)$$

The temperature term, T, is decreased over time using different approaches. These are known as schedules. For example, a linear schedule decreases the temperature using a linear calculation based on the length of time which has elapsed since the start of the search. Simulated annealing leads to the probability of accepting a change to the molecule decreasing over time and thus convergence upon a minimum in the search space. Due to the flexibility and ease of implementation, stochastic search with and without simulated annealing has been implemented in several de novo design software, including CONCEPTS (Pearlman & Murcko, 1993), CONCERTS (Pearlman & Murcko, 1996), MCDNLG (Gehlhaar et al., 1995), SMoG (DeWitte et al., 1997; DeWitte & Shakhnovich, 1996) and SYNOPSIS (Vinkers et al., 2003).

A natural data structure to randomly traverse chemical space is the Markov chain. The Markov chain is a connected graph where the edges are the transition probabilities. Stochastic search is then performed by sampling the transition probabilities to choose fragments to combine or delete. For example, FoG (Kutchukian et al., 2013) used a Markov chain structure to traverse chemical space to compounds of interest.

While stochastic search can certainly traverse huge areas of chemical space, care needs to be taken in constructing the probability distributions that are sampled. Moreover, the approaches inherently

require multiple searches to achieve consistent results. The approach of sampling probability distributions forms the basis of many deep neural network-based generative design algorithms (See 3.6).

3.5.1.1 Evolutionary Methods

An alternative approach to stochastic search is evolutionary inspired approaches to search. Evolutionary algorithms aim to find individuals in a population that satisfy a set of objectives (Vikhar, 2017). The methods are inspired by Darwinian evolution (Darwin, 1859). Individuals in a population are encoded as chromosomes. The encoded chromosomes are then mutated using genetic operators. A genetic operator is an operator applied to an individual in the population to cause a change to the individual. The main types of operators are mutation and crossover. Mutation applies a change to a single individual, such as addition of an atom or fragment. In contrast crossover swaps atoms and fragments from the population of individuals. Evolutionary algorithms rely on the following process to search through the chemical space:

1) Initialise population of individuals.
2) Score the individuals in the population.
3) Select the best-performing individuals from the population.
4) Apply operators to generate a new population.
5) Repeat steps 2) to 4) until an end criterion is reached.

The most popular evolutionary-based search approach is the genetic algorithm and several de novo design algorithms have used genetic algorithms as part of their search approach. One of the first published approaches to de novo design with genetic algorithms was Chemical Genesis which used 3D graph-based methods for structure representation (Glen & Payne, 1995). Moreover, the usage of graph-based representations for genetic algorithms have been explored as part of CoG (Brown et al., 2004), ACSESS (Rupakheti et al., 2015; Virshup et al., 2013), GB-GA (Jensen, 2019), GB-EPI (Verhellen & Van Den Abeele, 2020) and STONED (Nigam et al., 2021). Another early example is the SMILES-based evolutionary algorithm patented by (Weininger, 1995). The patented approach mutated SMILES strings to generate molecules that fit the desired property profile.

In 1996, a genetic algorithm-based approach was introduced that was coupled to a neural network (Devillers, 1996). The neural network was used to score the molecules and the genetic algorithm to search for better scoring compounds. The approach was recently reintroduced to the literature utilising a deep neural network for molecule scoring (Nigam et al., 2019).

In contrast to atom-based construction approaches, fragment-based methods can utilise additional genetic operators such as the grow and link operators in Ligbuilder (Wang et al., 2000; Yuan et al., 2011, 2020), AutoGrow (Durrant et al., 2009, 2013; Spiegel & Durrant, 2020) ADAPT (Pegg et al., 2001), PEP (Budin et al., 2001), GANDI (Dey & Caflisch, 2008).

Even with the rise in popularity of deep learning approaches (see section 3.6), genetic algorithms have demonstrated themselves as highly competitive in designing small molecules. Recently, in the GuacaMol benchmark (Brown et al., 2019), a popular de novo design benchmark designed to compare and contrast different de novo design software, GB-GA demonstrated exemplary performance out performing more computational expensive approaches (Jensen, 2019).

An alternative approach to using a genetic algorithm is genetic programming (Koza, 1994). Genetic programming (GP) utilises a tree-based structure to construct individuals which are then mutated following the same principles as genetic algorithms. A tree is a graph structure starting from a root node. GP utilises tree structures as an alternative to chromosomes. GP applies mutations and operations to the tree directly. For example, branches can be exchanged, or individual nodes in the tree can be replaced. Venkatasubramanian (Venkatasubramanian et al., 1995) introduced a limited approach to tree-based GP which was further improved by Nachbar (Nachbar, 1998, 2000). Nachbar used a tree of characters as the molecule representation. The mutation and crossover operations included changing nodes in the tree or swapping entire branches from other trees generated in the population. Nachbar also extended the GP approach to include ring-opening, ring-closing, and vicinal atom exchange. Compilation of the final molecule is obtained by simply traversing the tree structure. The main limitation of the approach by Nachbar was that it was written in Mathematica, and it did not generate complex structures.

A more recent example is the molecule evoluator (Lameijer et al., 2006). The molecule evoluator uses the same approach as Nachbar, except it utilises a modified SMILES notation known as tree SMILES. Tree SMILES introduced a linear notation derived from SMILES strings, allowing mutations and crossover to occur at specific substitution points in the molecule.

Further to evolutionary approaches are memetic approaches. Memetic approaches attempt to search a search space by mimicking a swarm of insects in a process known as swarm intelligence. Initially, a population of individuals is created. Then, the individuals are allowed to traverse the search space. If an individual in the population finds a region of interest, then that region's coordinates are communicated to the rest of the individuals in the population. The individuals then bias their traversals closer to the region of interest until, eventually, the population converges upon the desirable region. Swarm intelligence algorithms derive their traversal from different natural

swarms such as bees and ants. The most general approach to swarm intelligence is particle swarm optimisation (PSO). An example of a de novo design software utilising PSO is COLIBREE (Hartenfeller et al., 2008). COLIBREE uses a starting scaffold and identified attachment points. The attachment points act as particles. A molecule is then grown from each attachment point and then evaluated. If a good score is identified, this is then be communicated to other particles via a swarm memory which records high scoring fragments and linkers. The approach utilises a tree data structure to traverse the space. A more recent approach to PSO is Molecule Swarm Optimisation (MSO) (Winter et al., 2019). MSO traverses a latent space encoded via a deep neural network.

Another swarm intelligence approach explored for de novo design is the Molecular Ant Algorithm (MAntA) (Hiss et al., 2014; Reutlinger et al., 2014). MAntA is an ant colony optimisation approach (Dorigo & Blum, 2005; Dorigo & Di Caro, 1999). MAntA relies on the creation of trails in the chemical space mimicking ant's pheromone trails. The trails act as a biasing approach to force agents closer to a minimum. Ant colony optimisation relies on previous trails increasing in strength as more agents pass over the same region of space compared to particle swarm optimisation. Similar to COLIBREE, MAntA relies on fragment-based approaches to construct molecules.

3.5.2 Deterministic methods

Deterministic methods are search methodologies used to traverse the chemical space that rely on rule- or function-based approaches. Deterministic methods are generally less popular compared to stochastic counterparts due to chemical space being both large and exceptionally rough. Therefore, it is hard to build deterministic search algorithms that can move through the chemical space effectively. However, it is important to note that many early de novo design software relied upon deterministic search methods with the approach still being utilised.

Search is often constructed via the use of the search tree. A tree is a graph containing nodes and edges, starting from a root node. The nodes contain molecules, and the edges are transformations between molecules. Trees are a common part of de novo design algorithms. To grow a tree, two strategies exist breadth-first and depth-first search. Breadth-first scores all nodes and then enumerates the next level of the tree. This process is repeated until either an entire tree is grown, or a computational limit is reached. In contrast, a depth first search grows the tree to a particular depth, returns to a level and then descends again via a different branch. Both approaches have benefits when being used in a de novo design context.

The breadth-first search yields a complete tree in an enumerative approach where all molecules are expanded one step at a time. The main limitation of the breadth first search is that vast numbers of molecules are generated. Often an additional part of the breadth-first search is the inclusion of a

pruning step. Pruning is the removal of nodes from the search tree. Often pruning of the search tree is carried out by simply scoring the molecules and then selecting the top-performing molecules to expand next. This pruning approach can, however, lead to limited performance due to the presence of local optima along paths of sequential molecule construction steps. However, several de novo design methods have used the approach with some success, for example, HSITE (Danziger & Dean, 1989b), LUDI (Böhm, 1992a), NEWLEAD (Tschinke & Cohen, 1993), HOOK (Eisen et al., 1994), MCSS (Miranker & Karplus, 1991), RASSE (Luo et al., 1996), PRO_SELECT (Murray et al., 1997), BREED (Nisius & Rester, 2009), DOGS (Hartenfeller et al., 2012), Reaction vectors (Patel, 2009; Patel et al., 2009), DINGOS (Button et al., 2019), RENATE(Ghiandoni, 2019) and DESMILES (Maragakis et al., 2020).

In depth-first search, a tree is grown to a specific depth by selecting one node at each level. This prevents a combinatorial explosion from occurring, however, this can lead to poor quality solutions being generated if only the best scoring molecule is selected at each tree level. Depth-first is often combined with a backtracking approach. Backtracking is a process where local optima can be avoided by revisited earlier nodes which are expanded down to the correct depth. Several approaches to de novo design using depth-first search have been created, for example, 3D Skeletons (Gillet et al., 1990), Diamond Lattice (Lewis, 1990), SPLICE (Ho & Marshall, 1993), Genstar (Rotstein & Murcko, 1993a), Groupbuild (Rotstein & Murcko, 1993b), PRO_LIGAND (Clark et al., 1995), Dycoblock (Zhu et al., 2001) and F-Dycoblock (Zhu et al., 2001).

3.6 Generative design

In recent years, de novo design utilising deep learning-based methodologies has increased in popularity. Deep learning-based de novo design is often referred to as generative design. The popularity of generative design can be attributed to several factors including improvements in computer hardware; leveraging natural language processing research; increasing volume of available data; and the availability of open-source toolkits to develop deep neural networks. The improvements to computer hardware allowed for the development of deep neural networks utilising large scale parallel computation(Raina et al., 2009). These, in turn, led to significant improvements in the modelling of sequences in text-based representations for natural language processing (NLP) (Young et al., 2018). Generative design has benefitted from these improvements in NLP as molecules can be represented as SMILES strings which can be modelled as sequences. As NLP research has advanced, the deep neural network architectures popular within the NLP community have been adapted for molecular design tasks (Chen et al., 2018; Xu et al., 2019) . These networks are further fuelled by a large amount of available data across many public databases such as ChEMBL and ZINC (Davies et al., 2015; Irwin et al., 2020; Mendez et al., 2019; Sterling & Irwin, 2015) . Finally, as deep

learning has become more popular, this has also led to improvements in open source libraries which allow practitioners to build deep neural networks (Chollet et al., 2015; Google Research, 2015; Paszke et al., 2019; The Theano Development Team et al., 2016), thus allowing for the development of generative design specific architectures (Elton et al., 2019; Meyers et al., 2021). In this section, a description of methodologies utilising deep neural networks for de novo design is given.

The basic structure of a neural network (NN) comprises an input layer, a hidden layer, and an output layer. An input layer takes the inputs, a hidden layer is a set of transformations that occur to a set of inputs, and the output layer transforms the hidden layer into a recognisable output.

A deep neural network (DNN) is a more advanced version of a neural network and is characterised by having several hidden layers for data to pass through prior to the output layer (James et al., 2021). Hence a deep neural network comprises an input layer, a set of hidden layers and an output layer. The input layer takes the inputs, the first hidden layer transforms the inputs to a set of outputs which are passed to the second hidden layer which transforms them to outputs and so on until finally, the last hidden layer passes the outputs to the output layer which transforms them to a recognisable output. Many different configurations of deep neural networks exist; these are known as architectures (Elton et al., 2019).

In de novo design, the goal is for the DNN to output a molecule that fits a pre-defined property profile. Often generative design approaches utilise SMILES strings to represent output molecules. To allow a DNN to train and generate molecules effectively, SMILES strings are grown from a starting character to an ending character. During training, the DNN learns the relationship between a sequence of SMILES characters and the probability of generating the next SMILES character. Training data for generative design methods is often taken from pre-constructed molecule databases.

A fundamental building block of many architectures which are used in generative design approaches is the recurrent neural network (RNN)(Elton et al., 2019; Liu et al., 2019). An RNN takes an input and predicts two outputs; the first output is a single character, and the second output is the values from the hidden layer. The output character is saved and inserted into the next input layer. The hidden layer output is also inserted into the next hidden layer. This process repeats until the RNN generates an ending character. This is shown in the Figure 3-2 below:

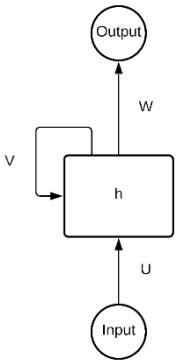


Figure 3-2 Folded RNN, U are the input weights, V is the weights of the hidden layer h, and W are the output weights. An input is provided such as a starting character and the RNN is sampled until an ending character is obtained.

RNN's are popular due to their relative ease of implementation and ability to learn from large amounts of training data. Extensions to RNNs include long short-term memory (LSTMS) cells, whereby memory cells are added to the network to prevent forgetting earlier learnt information. RNNs have been further improved by a process known as fine-tuning. Here an RNN is first trained on a collection of molecules and then refined on a smaller, more focused library of compounds with desired properties. The resulting RNN then can generate molecules that fit the requirements of the smaller library. Alternatively, a popular approach has been to utilise a tree search approach to prioritise SMILES fragments which are then completed using a trained RNN. The tree search approach used is the Monte Carlo Tree Search which will be described in Chapter 6.

Variational autoencoders (VAE) are an alternative and recently popular approach to generative de novo design and are based on Autoencoders (AE). AEs rely on an encoding and then decoding process, as is shown in Figure 3-3. Molecules are passed into the encoder and then decoded by the decoder. The goal of training is then to minimise the error between input and output. Moreover, by training an AE, the hidden layers form a latent space between the encoder and decoder. The latent space is then sampled, and new molecules are generated. VAEs take a probabilistic approach to encoding and decoding the inputs and outputs. Several different approaches to VAEs have been created such as ChemVAE (Gómez-Bombarelli et al., 2018), GraphVAE (Simonovsky & Komodakis, 2018) and SD-VAE (Dai et al., 2018).

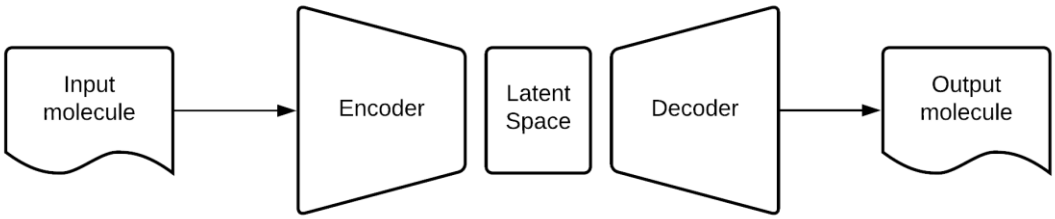


Figure 3-3 An input molecule is passed into the encoder and is transformed via a latent space and decoded by the decoder. During training the goal is to minimise the differences between input and output

Deep reinforcement learning (DRL) is an approach to generating SMILES strings by training an agent which interacts with an environment. The agents' goal is to maximise the sum of rewards by choosing specific actions. In a generative DRL approach, the actions are SMILES characters. An agent, which is usually an RNN, is trained to predict the probable reward of each SMILES character based on a list of input characters. Chosen SMILES characters are stored in the list, and a reward is generated via Monte Carlo sampling utilising the REINFORCE (Williams, 1992) algorithm (see section 5.7.1). The agent is given the sampled reward, and the RNN is updated. The process then repeats until the list of characters is full, and a complete molecule is formed. The generative approach ReLeaSE (Popova et al., 2018) utilised the REINFORCE algorithm to successfully design small molecules of interest. Additional reinforcement learning generative design approaches include REINVENT (Olivecrona et al., 2017) utilising a Bayesian loss function, and DrugEx (Liu et al., 2019) which introduced an additional exploration term.

Generative adversarial networks (GAN) are DNN architecture split into two components: a generator and a discriminator. The generator generates molecules, and the discriminator attempts to distinguish the molecules generated by the generator from actual molecules provided from a dataset. The output from the discriminator is passed to the generator. Over sequential training iterations, the generator improves such that the discriminator cannot distinguish generated molecules from the real dataset examples. The methodology is similar to the approach in 1996 by (Devillers, 1996), which relied on a genetic algorithm coupled with a neural network. Several approaches to GANs have been developed for generative design including ORGAN (Guimaraes et al., 2017) and MolGAN (De Cao & Kipf, 2018).

The crucial limitation of all the approaches described so far is that synthetic accessibility is not considered. Recently, deep learning approaches have been introduced that attempt to utilise reaction-based de novo design. These approaches use DNNs coupled with reaction data to build molecules sequentially. Several approaches utilise Deep RL approaches to select reactions, such as REACTOR (Horwood & Noutahi, 2020), ChemBO (Korovina et al., 2019), and Apollo 1060 (Gottipati et

al., 2020). Alternatively, methods such as Molecule Chef (Bradshaw et al., 2019) and synthesis directed acyclic graph generators (Bradshaw et al., 2020) rely on more complex representations.

Finally, a notable reaction-based deep learning method is the DINGOS approach (Design of innovative NCEs generated by optimisation strategies) (Button et al., 2019). DINGOS uses a DNN approach to select building blocks from an extensive collection of molecules. The building blocks were then combined using reaction templates. DINGOS was used as part of a prospective de novo design study with molecules designed, synthesised, and tested for biological activity.

Finally, the output of a DNN is dependent on the training data of the DNN. For example, the choice of SMILES strings, while efficient, limits the creation of complex ring systems or heterocyclic structures. Additionally, DNN methods are "blackbox" approaches and therefore it is hard to interpret why a DNN chose to generate a specific SMILES character. Attempts utilising reaction-based templates are a promising direction to overcoming these limitations. However, these methods rely on large amounts of reaction training data and extensive building block collections.

3.7 Validation of de novo design methods

Testing and validation of de novo design software is crucial to understanding its practical utility. Broadly, testing of de novo design is split into two categories retrospective validation and prospective validation. Retrospective validation is where the aim is to recreate a known molecule often as part of a set of rediscovery problems. Prospective validation is when a scoring function is defined, the molecules are designed by the de novo design software and then made and tested in the lab. Prospective validation requires extensive resources.

Additionally, comparative studies between de novo design software are difficult to achieve for several reasons, for example, scoring functions are not explicitly defined, data is often not made available, and only the best-generated molecules are published. Finally, many early de novo design software is not publicly available, and therefore cross-comparison is difficult. This has led to difficulties due to survivorship bias and reproducibility bias, as frequently only positive results in de novo design are published, and many methods are not reproducible.

Recently with the advent of open science (Murray-Rust, 2008; Woelfle et al., 2011) there has been a concerted effort to improve code availability and benchmarking approaches (Brown et al., 2019; Polykovskiy et al., 2020). A notable suite of de novo design problems that has become a popular benchmark is the GuacaMol benchmark (Brown et al., 2019). GuacaMol is a set of de novo design problems that have been created along with a set of training data to compare modern generative de novo design approaches. Two types of problems exist in benchmark goal-directed and distributional

learning. Goal-directed learning attempts to design molecules that fit a predefined property profile. In comparison, distributional learning aims to generate a distribution of molecules with a specific property. A GuacaMol leaderboard has also been created to compare different de novo design approaches.

3.8 Conclusion

This chapter has outlined the history of de novo design, from the 1980s to modern generative design. The chapter has focused on three key areas: molecule scoring, molecule construction, and molecule search and optimisation. A fundamental limitation of many de novo design approaches identified in this chapter has been molecules having limited synthetic tractability.

Reaction-based de novo design is a promising direction to overcome this limitation via synthetic transformations encoded from literature synthetic schemes. RV-based de novo design is an established approach to reaction-based de novo design. However, a key limitation of RV-based search is the over-reliance on complete enumeration via breadth first search and issues with sequential application of RVs due to the presence of local optima. The next chapter describes RV-based de novo design as a form of molecule construction in some detail due to it forming the foundation of the methods developed in this thesis.

Chapter 4 Reaction Vectors

Reaction vectors (RVs) encode structural transformations of starting materials (SMs) to products based on the difference between SM descriptors and product descriptors (Broughton et al., 2003). The following equation is often used to describe an RV:

$$RV = P - (SM + R) \quad (6)$$

Where RV is the reaction vector which is a collection of descriptors encoding the transformation between SM and product, SM represents descriptors present in the starting material. R is the descriptors present in the reagents. Finally, P is the descriptors present in the products.

The descriptors in RVs are based upon atom pairs. An atom pair (AP), is a topological molecular descriptor that encodes a pair of atoms separated by a given topological distance. Atom pairs can vary depending upon the number of bonds between the corresponding atoms. For example, AP2 describes a pair of atoms joined by a bond, AP3 describes a pair of atoms separated by two bonds. Atom pairs are ideal for usage in reaction encoding due to the richness of bond and atom information stored in the descriptor.

In the latest implementation of RVs extended from (Patel, 2009; Patel et al., 2009), a custom AP descriptor is used of the following form:

$$X_1(h_1, p_1, r_1) - S(BO) - X_2(h_2, p_2, r_2) \quad (7)$$

Where $X_1$and X_2 are the corresponding atom typing of the two atoms. h_1 and h_2 represent the number of connections to non-hydrogen atoms that are attached to atom one and two, respectively. $p1$, and $p2$ are the number of pi electrons shared by the atom and is determined based on the bond order of the bonds connecting to the atom. r_1 and r_2 is the ring membership count of atoms one and two respectively. S is the separator indicating the number of atoms. Finally, BO describes the corresponding bond order between the two atoms for AP2 descriptors only. Single bonds have a BO of one; double bonds have a BO of two, triple bonds have a BO of three; and aromatic bonds have a bond order of four. No explicit stereochemistry between the atoms nor any explicit hydrogens are described using the modified AP descriptor therefore stereochemistry is not considered in the RV framework.

A simple example of computing all AP2 descriptors using the custom AP descriptor shown in Table 4-1

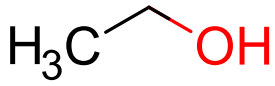
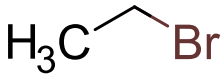
Molecule	Unique atom pairs with their occurrences
	C(2,0,0)-2(1)-C(1,0,0)=1 O(1,0,0)-2(1)-C(2,0,0)=1
	Br(1,0,0)-2(1)-C(2,0,0)=1 C(2,0,0)-2(1)-C(1,0,0)=1

Table 4-1 Example showing two simple examples of AP2 descriptors for ethanol and bromoethane

A RV is computed by subtracting the atom pairs of the reaction(s) from those of the product(s) and can result in negative and positive counts of atom pairs. When more complex reaction transformations are encoded in the RV formalism, the number of atom pairs can increase significantly.

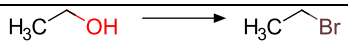
Reaction	Negative AP2	Positive AP2	Reaction Vector AP2
	O(1,0,0)-2(1)-C(2,0,0)	Br(1,0,0)-2(1)-C(2,0,0)	Br(1,0,0)-2(1)-C(2,0,0)=1 O(1,0,0)-2(1)-C(2,0,0)=-1

Table 4-2 Example of a AP2 RV and corresponding negative and positive AP2s for a simple substitution reaction

RVs for structure generation are encoded as a list of AP2 and AP3 descriptors (Patel, 2009; Patel et al., 2009). An example of a ring-closing reaction described in the RV formalism is shown in Table 4-3 below. The ring-closing reaction highlights the complexity that can arise when encoding RVs.

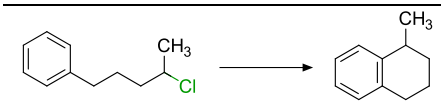
Reaction	Negative AP2 and AP3	Positive AP2 and AP3	Reaction Vector
	C(2,0,0)-2(1)-C(2,0,0)	C(2,0,0)-2(1)-C(2,0,0)	C(2,0,0)-2(1)-C(2,0,0)=-2
	C(2,0,0)-2(1)-C(2,0,0)	C(2,0,0)-2(1)-C(2,0,0)	C(2,0,1)-2(1)-C(2,0,1)=2
	C(2,1,1)-2(1)-C(2,1,1)	C(2,1,1)-2(1)-C(2,1,1)	C(2,1,1)-2(1)-C(2,1,1)=-1
	C(3,0,0)-2(1)-C(1,0,0)	C(3,0,0)-2(1)-C(1,0,0)	C(3,0,0)-2(1)-C(1,0,0)=-1
	C(3,0,0)-2(1)-C(2,0,0)	C(3,0,0)-2(1)-C(2,0,0)	C(3,0,0)-2(1)-C(2,0,0)=-1
	C(3,1,1)-2(1)-C(2,0,0)	C(3,1,1)-2(1)-C(2,0,0)	C(3,0,1)-2(1)-C(1,0,0)=1
	C(3,1,1)-2(1)-C(2,1,1)	C(3,1,1)-2(1)-C(2,1,1)	C(3,0,1)-2(1)-C(2,0,1)=1
	C(3,1,1)-2(2)-C(2,1,1)	C(3,1,1)-2(2)-C(2,1,1)	C(3,1,1)-2(1)-C(2,0,0)=-1
	Cl(1,0,0)-2(1)-C(3,0,0)	Cl(1,0,0)-2(1)-C(3,0,0)	C(3,1,1)-2(1)-C(2,1,1)=-1
	C(2,0,0)-3-C(1,0,0)	C(2,0,1)-3-C(1,0,0)	C(3,1,1)-2(2)-C(2,1,1)=-1
	C(2,0,0)-3-C(2,0,0)	C(2,0,1)-3-C(2,0,1)	C(3,1,2)-2(1)-C(2,0,1)=1
	C(2,1,1)-3-C(2,0,0)	C(2,1,1)-3-C(2,0,1)	C(3,1,2)-2(1)-C(2,1,1)=2
	C(2,1,1)-3-C(2,0,0)	C(3,0,1)-3-C(2,0,1)	C(3,1,2)-2(1)-C(3,0,1)=1
	C(2,1,1)-3-C(2,1,1)	C(3,0,1)-3-C(2,1,1)	C(3,1,2)-2(2)-C(3,1,2)=1
	C(2,1,1)-3-C(2,1,1)	C(3,1,2)-3-C(1,0,0)	Cl(1,0,0)-2(1)-C(3,0,0)=-1
	C(3,0,0)-3-C(2,0,0)	C(3,1,2)-3-C(2,0,1)	C(2,0,0)-3-C(1,0,0)=-1
	C(3,1,1)-3-C(2,0,0)	C(3,1,2)-3-C(2,0,1)	C(2,0,0)-3-C(2,0,0)=-1
	C(3,1,1)-3-C(2,1,1)	C(3,1,2)-3-C(2,0,1)	C(2,0,1)-3-C(1,0,0)=1
	C(3,1,1)-3-C(2,1,1)	C(3,1,2)-3-C(2,1,1)	C(2,0,1)-3-C(2,0,1)=1
	Cl(1,0,0)-3-C(1,0,0)	C(3,1,2)-3-C(2,1,1)	C(2,1,1)-3-C(2,0,0)=-2
	Cl(1,0,0)-3-C(2,0,0)	C(3,1,2)-3-C(2,1,1)	C(2,1,1)-3-C(2,0,1)=1
		C(3,1,2)-3-C(2,1,1)	C(2,1,1)-3-C(2,1,1)=-2
		C(3,1,2)-3-C(3,0,1)	C(3,0,0)-3-C(2,0,0)=-1
			C(3,0,1)-3-C(2,0,1)=1
			C(3,0,1)-3-C(2,1,1)=1
			C(3,1,1)-3-C(2,0,0)=-1
			C(3,1,1)-3-C(2,1,1)=-2
			C(3,1,2)-3-C(1,0,0)=1
			C(3,1,2)-3-C(2,0,1)=3
			C(3,1,2)-3-C(2,1,1)=4
			C(3,1,2)-3-C(3,0,1)=1
			Cl(1,0,0)-3-C(1,0,0)=-1
			Cl(1,0,0)-3-C(2,0,0)=-1

Table 4-3 Table of positive and negative AP2 and AP3 descriptors encoded in the RV formalism.

It is important to note that a RV is not explicitly aware of incorrect chemistry. That is, the RV is computed solely on the AP2 and AP3 descriptors. Therefore, it is at the user's prerogative to appropriately clean reactions before encoding the RV.

The RV encodes the reaction centre by using the AP2 and AP3 descriptors to encode the transformation. In the framework of reaction representation, an RV is a data-driven approach to reaction representation. One of the main usages of RVs has been to use RVs as a knowledge-based molecule construction approach.

4.1 Molecule construction with reaction vectors

Molecule construction with RVs (structure generation) has been extensively explored in recent years. Moreover, the RV implementation has also improved over subsequent years since the original approach in 2009 (Patel, 2009). Structure generation is performed using RVs by first removal and then addition of AP2 and AP3 descriptors from a reaction and optional reagent. The first approach of using RVs for molecule construction was explored and validated by retrospective validation (Patel, 2009; Patel et al., 2009). In this case, a set of reactions where the starting molecule and products were known was used to encode a set of RVs. The RVs were then applied to the starting molecules to observe whether the product could be regenerated.

Moreover, Patel also investigated using RVs for structure generation in various de novo design scenarios. Following the validation success, further work was carried out to improve the structure generation algorithm and demonstrated that a list of AP2 and AP3 descriptors was an adequate balance of encoded information and generalisability (Hristozov et al., 2011). Additional applications have been explored since then including, de novo design using Pareto ranking (Gillet et al., 2013), reaction sequence vectors (Wallace, 2016) and RENATE (pseudoRetrosynthEtic desigN using reAcTion vEctors) (Ghiandoni, 2019). Due to their extensive use throughout this work, it is useful to provide an overview both of the original method of structure generation proposed by (Patel et al., 2009) and the more efficient approach to structure generation (Hristozov et al., 2011) used in this work.

4.1.1 Structure generation prerequisite components

To be able to use RVs for structure generation, three components are required. A SM, a set of reagents, and a collection of RVs. The SM is the molecule that will be transformed. The reagents are a list of molecules (can be empty) that can be combined with the corresponding SM. The collection of RVs is a set of RVs that have been previously collected from other reaction examples. The RVs are stored in a RV database, which can store additional metadata.

4.2 Structure generation – original approach

As highlighted previously, structure generation can be effectively described as removing negative atom pairs and adding positive atom pairs to the SM. In the original approach (Patel, 2009), the molecule is first broken down using the negative atom pairs. The negative fragmentation (decomposition) yields a fragment with additional atom metadata used to describe the fragmentation points. The positive atom pairs are then added. Unfortunately, there are many ways to add additional atom pairs back to the decomposed fragment. Hence, a tree of possible additions is created and is searched using the breadth-first search algorithm. Molecules are heuristically

removed from the tree if they contain atom pairs that are not present in the expected product. It is important to note that the number of molecules generated during the addition of fragments can be more than one. A schematic of the original algorithm in Figure 4-1 below shows a simple halogen interconversion reaction. A more complex example of an epoxide ring-opening is shown in Figure 4-2. The steps are as follows; the molecules negative atom pairs are removed from the molecule based on the RV (blue above the reaction arrow). Attachment points are denoted by an asterisk (*). The positive atom pairs are then added (shown in red above the reaction arrow). The molecule(s) generated are then checked against the expected list of AP3 descriptors. If the molecule contains all expected AP3 descriptors, it is considered valid. For example, in Figure 4-2, two possible products could be generated. However, only the benzodiazol-6-ol contains the appropriate AP3 descriptors. Finally, it is essential to note that the addition of AP2s occurs sequentially, with many different possible structures needing to be traversed based on the bond formation process.

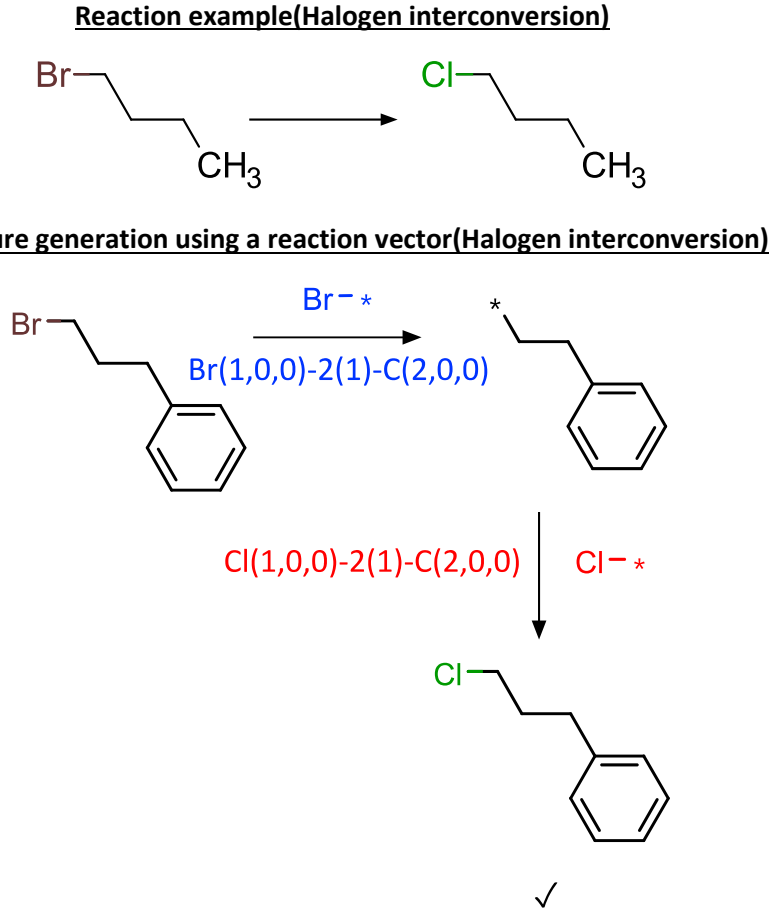
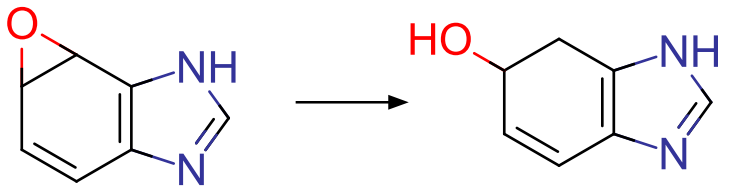


Figure 4-1 Simple halogen interconversion reaction using RVs. The blue fragment is the substructure removed and is encoded by the negative AP descriptors. The red fragment is the substructure added and is encoded by the positive atom pairs.

Reaction example (epoxide ring-opening)



Structure generation using a reaction vector (epoxide ring-opening)

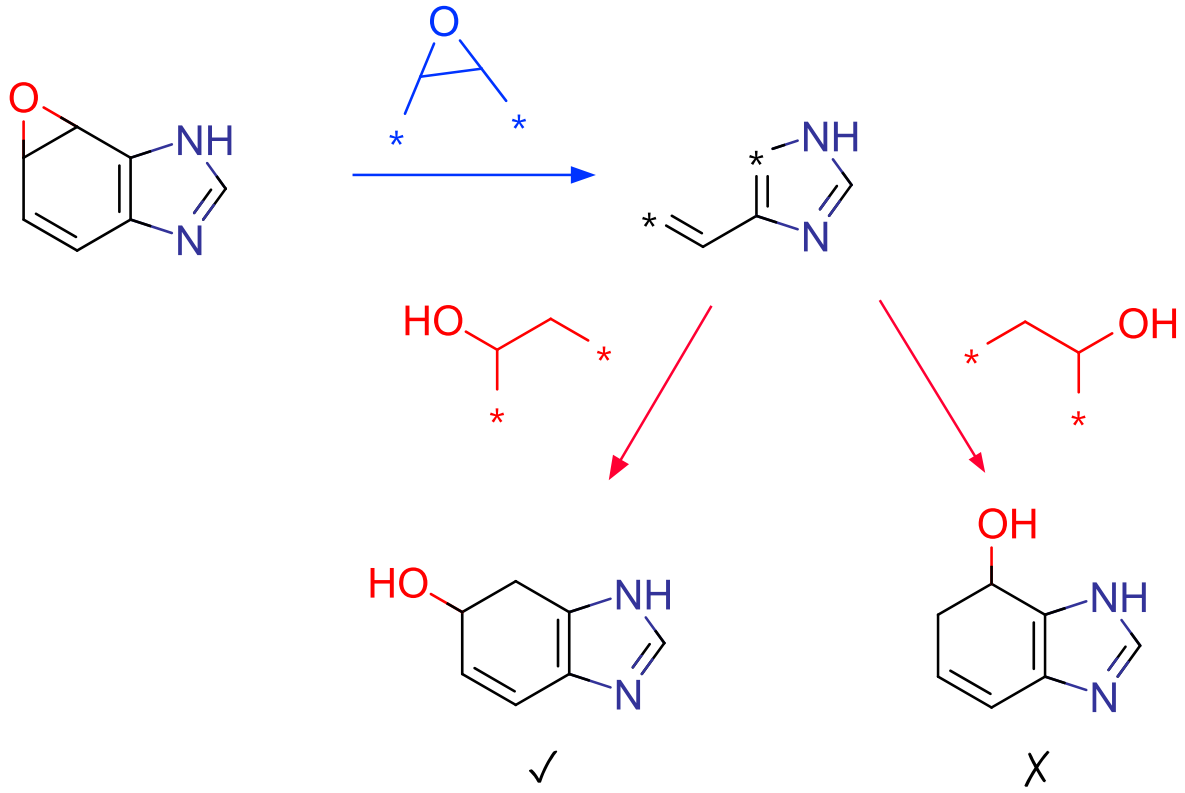


Figure 4-2 Epoxide ring-opening using a RV based on the original structure generation approach by (Patel, 2009; Patel et al., 2009). *The blue substructure is the AP2 descriptors to be removed. The red is the AP2 descriptors that will be added. In this example, it is important to note that the AP2 descriptors can be added in two different ways. Yielding either benzodiazol-6-ol(left) or benzodiazol-7-ol(right). benzodiazol-7-ol does not contain the expected AP3 of the products and is therefore rejected (cross mark).Example adapted from* (Ghiandoni, 2019)

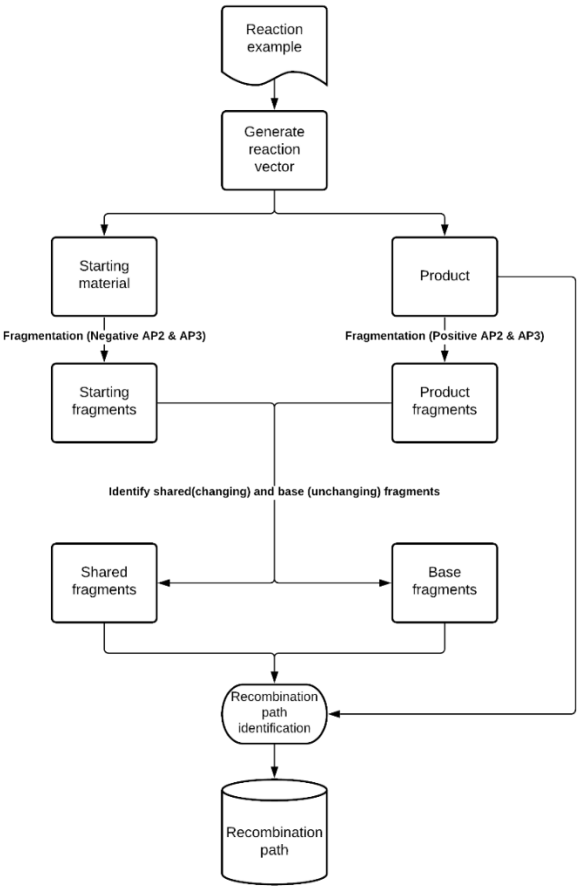
Remarkably, even utilising a near exhaustive search, the original approach of the RV structure generation yielded exemplary performance on validation again 5.6k reactions (85% successful validation). However, if more complex molecules or reaction centres are used, the original approach slowed down substantially, in part due to the large number of possible products which could be generated from extensive positive AP2 descriptors. Hence, several improvements were made to improve the overall performance of the RV approach.

4.3 Reaction vectors-current implementation

As highlighted above, the core premise of a structure generation using RVs is simply removing and adding APs. The original approach utilised a simple search approach to traverse the bond making and breaking tree. However, to improve the overall performance of the structure generation, additional information was stored within the RV database (Hristozov et al., 2011). The additional information is a numbered list of substructural fragments which is known as a recombination path.

To generate a recombination path, the following procedure is carried out. First, for a given reaction, calculate its RV. Then, apply the RV to SMs and products. Applying a RV in this context means fragmenting the SM's structure based on the negative AP2 and AP3 descriptors. Likewise, applying a RV in this context means fragmenting the product's structure based on the positive AP2 and AP3 descriptors. This fragmentation process is termed "reverse fragmentation". Following reverse fragmentation, two sets of fragments are generated: a starting set of fragments and a product set of fragments. The fragments are then partitioned into two groups, a changing set of fragments called the shared fragments, and a set of unchanging fragments called the base fragments. First, the largest unchanging fragment is determined using a maximum common substructure algorithm. Next, the largest unchanging fragment is removed from the product, resulting in a separate fragment called the reacting fragment. The product molecule, largest unchanging fragment and reacting fragment are then used to find a sorting of the shared fragments such that the product molecule can be created from the largest unchanged fragment. If an order can be determined, the sorted list of shared fragments and the largest unchanged fragment is stored, known as the recombination path. A schema of the fragmentation – recombination approach is shown in Figure 4-3 below.

Recombination path generation



Recombination path identification

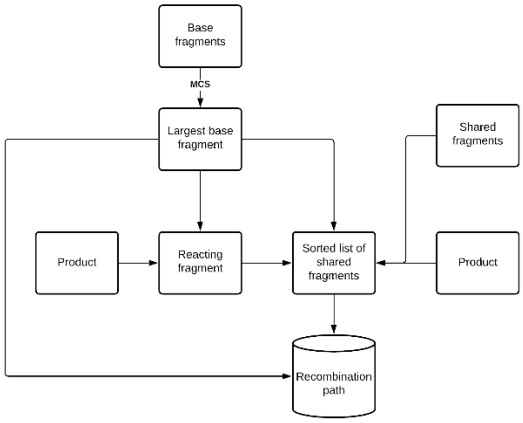


Figure 4-3 Process of generating a recombination path from a given reaction example. The figure is split into two parts the general recombination path schema, then a more detailed recombination path identification. First, two sets of fragments are generated, a base set and a shared set. The identification of the recombination path then begins (second schema). The base fragment is identified via an MCS. The MCS is removed from the product yielding the reacting fragment. Next, the list of shared fragments is sorted to give the reacting fragment starting from the base fragment and finishing with the product. This sorted list is stored as the recombination path along with the base fragment.

An important consideration when generating recombination paths is how to handle failure during the generation process. Examples of failure in the generation process are no-correspondence between fragment sets, or the reacting fragment cannot be regenerated from the base fragment to the product. If a failure occurs during the recombination path generation process, the original reaction vector algorithm is used instead. If recombination path generation and the original reaction vector algorithm fail, the reaction is discarded and not stored in the reaction vector database.

The generation of recombination paths was validated against the same dataset of reactions as the original algorithm (Hristozov et al., 2011). The validation resulted in 89% of the reactions being successfully encoded. However, the distribution of reaction classes encoded by the new recombination path approach differed. Notably, the recombination path's retrieval speed was orders of magnitude quicker than the original algorithm (0.015s vs 3.1s).

When structure generation is applied in a prospective manner using the recombination path approach the SM is fragmented based on the negative AP2 and AP3 descriptors contained in a RV. The remaining fragment is then combined with the sorted list of fragments stored in the recombination path in the RV database. This overcomes the expensive traversal of generating products while applying the positive AP2s described in the original approach (Patel et al., 2009). Following structure generation, a cleaning process is applied using the RDKit (*RDKit*, 2021). The cleaning process involves sanitising the molecule by checking for incomplete bonds, incorrect valences, or issues with aromaticity. If the generated molecule fails during the RDKit sanitisation process, the molecule is rejected.

It should be noted that a single molecule may have multiple points on the molecule, where a RV could be applied. Therefore, multiple possible reaction centres lead to a relationship between the SM and RVs. That is, one SM can have many RVs applicable to it.

An example of structure generation using a recombination path is shown in Figure 4-4.

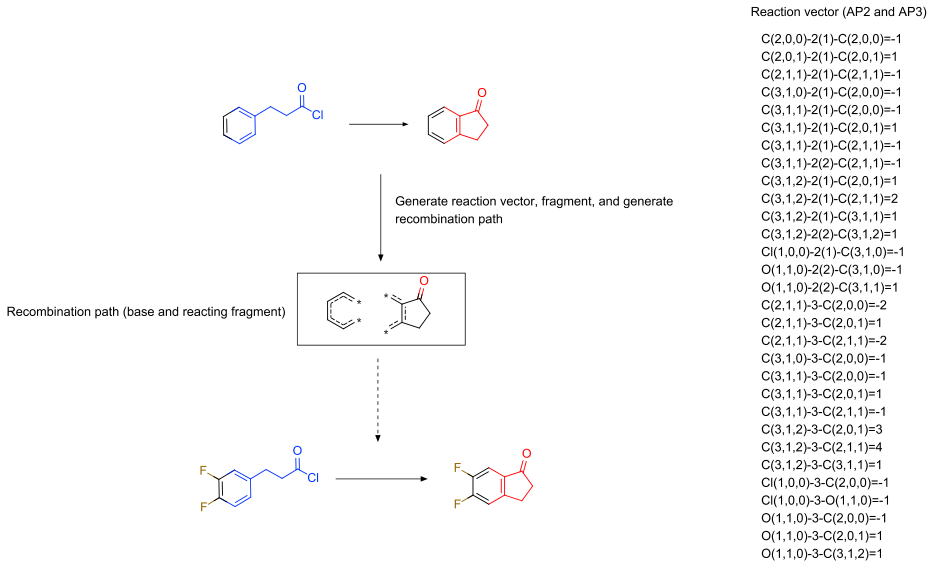


Figure 4-4 A recombination path is generated based on the RV. To apply the RV to a new molecule, the SM is fragmented based on the negative AP2 and AP3. The reacting fragment shown here is stored as a sorted list of fragments which is sequentially added to the molecule left over from fragmenting the SM. Adapted (Ghiandoni, 2019; Wallace, 2016)

4.3.1 Multi-reagent reactions

So far, only single SM transformations have been described. In practice, however, chemical reactions typically involve a SM and a reagent as a second molecule. The process of including multiple molecules is straightforward based on the recombination path approach. The negative AP descriptors are computed for all molecules on the left-hand side of the reaction arrow to use multi-reagent reactions. The RV is therefore stored with negative AP descriptors of all reagents. The corresponding recombination path is generated and stored using the process shown in Figure 4-3. When using a RV during structure generation, the negative AP descriptors in the RV are removed from the SM. If all negative AP descriptors in the RV are not found in the SMs, then the reagent pool is queried for molecules containing those missing atom pair descriptors. The process of using the recombination path is, however, unchanged.

4.4 Additional improvements

Since the improved RV algorithm was proposed in 2011, there have been minor improvements over time. These improvements include indexing the RV database to improve retrieval of AP descriptors and improved hashing to better match fragments during the structure generation process (Wallace, 2016). Finally, the database's metadata has also improved with reaction classes based on the SHREC ontology (Ghiandoni et al., 2019).

4.5 Conclusions

This chapter has described the original (Patel, 2009; Patel et al., 2009) and current RV implementations (Hristozov et al., 2011). RVs are the difference between product and reagent atom pair descriptors. By utilising the atom pair approach, the reaction centre and a surrounding region could be encoded. Storage and retrieval of RVs allowed RVs to be applied prospectively as part of a structure generation process. Moreover, improvement to the RV approach has facilitated the use of ever-larger collections of reactions and reagents (Ghiandoni, 2019).

Finally, de novo design using RVs as a structure generation method has been a long term research focus (Ghiandoni, 2019; Gillet et al., 2013; Patel, 2009; Wallace, 2016). The main limitation of these prior approaches has been an over-reliance on a "full enumeration". A full enumeration is a naïve brute force search approach used during RV-based de novo design. Full enumeration is an exhaustive application of every applicable RV and corresponding applicable reagent. Thus, full enumeration is susceptible to a combinatorial explosion of possible products. Moreover, as reaction collections have increased, the usage of the full enumeration approach has led to a substantial decrease in the utility of RVs for de novo design. Hence, in the next chapter, machine learning and the reinforcement learning are overviewed to build a foundation which can be used to overcome the limitation of the “full enumeration” approach.

Chapter 5 Machine learning

5.1 Introduction

Machine learning (ML) is a subfield of computer science that aims to learn patterns in data rather than rely on rules from domain experts (Murphy, 2012). ML has exploded in popularity due to the increase in the amount of data available, the development of computer hardware that has facilitated massively parallelised calculations, and improved computational methods(Bosc et al., 2019; Davies et al., 2015; Mendez et al., 2019; Raina et al., 2009). For example, utilisation of Graphical Processing Units (GPUs) coupled with parallelised computation led to the rise of deep learning, enabling improved performance across a wide range of domains and tasks (Chen et al., 2018; Lo et al., 2018; Schneider et al., 2020).

In chemistry, ML has become ubiquitous in both academia and industry (Schneider et al., 2020). This broad adoption is attributable to the large amount of chemical data that is now being generated, the availability of improved computer hardware and open source machine learning software libraries (Vamathevan et al., 2019).

Due to the size of the ML field, this chapter focuses on ML approaches that are directly relevant to this work: First, a basic overview of ML is given, including training, testing of ML algorithms. Second, the chapter discusses using large-scale ML, a subfield of ML dedicated to learning on large datasets. Third, an overview of validation of machine learning models is presented. Fourth, reinforcement learning (RL), focussing on single- and multi-step RL is presented. Finally, the chapter finishes with an overview of dimensionality reduction techniques.

The rationale for focusing on large scale ML and RL is based on the following reasons. Firstly, machine learning allows for approximating computationally expensive functions via learnt models which are inexpensive to evaluate. This technique is employed in Chapter 8 for computing values for large numbers of molecules.

Second, de novo design often generates vast numbers of molecules which are then represented as feature vectors with many dimensions. The large number of dimensions coupled with the considerable size of datasets leads to issues with memory during training. Large scale ML is employed later in the thesis as a solution to this memory issue.

Finally, RL is described as RL is primarily concerned with selecting optimal actions from a set of applicable actions. The selection of optimal actions is particularly pertinent for reaction vector-based de novo design, where the number of actions per molecule is very large.

5.2 Machine learning overview

The goal of machine learning (ML) is to learn a function that maps inputs to labelled outputs (Murphy, 2012). The learnt mapping between inputs and outputs is known as a model and the process of learning the mapping is known as training. Given a series of inputs, the model will approximate the values of the true labels. The process of approximating based on input data is known as a prediction(Murphy, 2012).

To train a model, a series of inputs is provided as training data, each input is labelled with a ground truth. The labelled training data is then passed into a machine learning algorithm. An ML algorithm is used to learn the model. Following training, a series of unlabelled data is passed into the learned model, and predictions are made for each input. The predictions are compared to the ground truth in a step known as testing which relies on the use of testing metrics to provide a numeric score by which the learnt model can be assessed. It is hoped that given a collection of training data, a machine learning algorithm will learn a function that generalises well enough such that the quality of the predictions is high (Murphy, 2012).

Several different categories of mappings between inputs and outputs can be used. The categories are classification, regression, and ranking. If a predicted value is a continuous value, the task is known as a regression task or simply regression (Hastie et al., 2001). If the output is a discrete or binary variable, the task is a classification task (Murphy, 2012). Finally, if the output is a series of rankings, the task is known as a ranking task (Alpaydin, 2014). ML models can differ further depending upon the number of tasks that need to be carried out. If a single input is mapped to a single output, it is known as a single task. If a single input is mapped to multiple outputs, then the task is known as multitask (Ruder, 2017b).

5.3 Regression

Regression takes a set of inputs and assigns a continuous output value to those inputs. Thus, regression requires ground truth values to learn a mapping between inputs and outputs. In chemistry, for example, regression tasks include predicting melting points of molecules or bioactivity predictions such as pIC50 (Bosc et al., 2019; McDonagh et al., 2015).

An input to a regression model is described using a set of features. For example, in the problem of bioactivity prediction using pIC50s, the input representation is a set of molecular features and the mapping between the features and bioactivity is learnt (Butler et al., 2018). In practice, multiple different input examples need to be provided with corresponding output values. Different input examples allow the model to learn a robust function which in the context of regression means that the function learnt is neither overfitted nor underfitted (Hastie et al., 2001). Fitness measures how

close a regression model can predict the ground truth value given a previously unseen input. If a regression model is overfitted, then the model will predict well only on the training data and poorly on unseen input data. In contrast, if a regression model is underfitted, then the model cannot predict well on either the training or unseen input data. To prevent overfitting strategies exist such as, having a diverse set of training examples or regularisation (Gong et al., 2019; Hastie et al., 2001). For example, a large, diverse set of molecules with annotated training values would be preferred over a small set of closely related molecules with annotated training values. Likewise, regularisation relies on penalisation of the fitted models coefficients. A further discussion of regularisation is provided in section 5.4.1.

5.4 Machine learning algorithms

Algorithms which learn regression functions are many and varied (Hastie et al., 2001). Two key categories of regression are linear and non-linear regression. In linear regression, the learnt function is a linear mapping between inputs and outputs weighted by coefficients. These coefficients are determined during the training process (James et al., 2021). In non-linear regression, more complex approaches are used to map between inputs and outputs. Both, linear models and non-linear models have been explored in chemistry. Linear models have been used extensively in various applications, such as modelling via Free-Wilson analysis (Free & Wilson, 1964) , whereby relationships across analogue series can be determined. In contrast bioactivity prediction or synthetic tractability prediction relies on non-linear approaches such as decision tree based methods including gradient boosted decision trees and random forest(Sheridan et al., 2016; Svetnik et al., 2003; Thakkar et al., 2021).

The section below describes several approaches utilised for regression modelling.

5.4.1 Linear regression

One of the most straightforward approaches to regression is to fit a linear function to a data set. Linear regression models are easily interpretable, easy to train on modest hardware, and have a myriad of different extensions (Agarwal et al., 2014). Figure 5-1 is an example of simple linear model. During training, the goal is to minimise the distance between the points and the linear function. The distance between a point and the linear function is called the residual. The goal is to minimise the error which is equivalent to minimising the sum of the squared residuals. If a small number of data points are available and the number of dimensions is low, then the best line can be found analytically using a process known as least squares (Hastie et al., 2001). If, however, the dataset is too large to fit into memory then a method known as stochastic gradient descent is required(Bottou

et al., 2018; Robbins & Monro, 1951; Ruder, 2017a) . Moreover, linear regression can be further extended with the introduction of better feature processing and regularisation. Regularisation introduces a penalty term in addition to the loss function (Murphy, 2012). Two different approaches to regularisation exist L1, and L2. L1 regularisation relies on the summation of absolute values of the coefficients of the fitted model (Tibshirani, 1996). Whereas L2 relies on the summation of the squared coefficients of the fitted model (Ng, 2004; Nigam et al., 1999). Additionally, the amount of penalisation can be tuned using a lambda coefficient parameter and requires tuning to find an optimal value. In both cases by penalising the fitted model based on its coefficients it minimises the influence of irrelevant features thus preventing overfitting. Finally, a further methodology to avoid overfitting in sparse domains is to only use values observed as non-zero. In this case only non-zero values are used as part of the fitted model and the rest are simply ignored.

Due to the importance of linear regression in this work a description of advanced approaches to linear regression on large datasets is provided (see section 5.5).

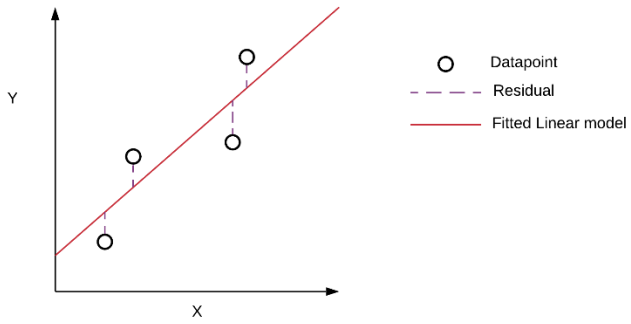


Figure 5-1 Simple linear model

5.4.2 Decision trees

A decision tree encodes a collection of rules(Quinlan, 1986). The rules are in nodes in the directed tree. The root node is the starting rule, and the tree is grown by adding rules to partition the data. The terminal nodes (leaves) are output values based on the average of each partition. During training, the collection of rules is constructed such that the sum of the square residuals is minimised. Due to the sequential nature of the applied rules, decision trees can encode non-linear relationships. Further improvements can be made depending on the method by which new rules are added to the tree (James et al., 2021).

Additionally, the approach of utilising decision trees can be further extended into forest approaches. Here multiple trees are grown, and the average of the resulting leaves are taken. These forest

approaches include random forests, whereby a large collection of trees are grown utilising random splitting and then pruning (Breiman, 2001). Alternatively, gradient boosted decision trees further improve the basic decision tree approach by sequentially growing trees utilising information from the previous tree (Freund & Schapire, 1999). Both approaches have seen significant interest in the process of building quantitative structure-activity relationship models for bioactivity prediction.

There are, however, several limitations to tree-based approaches. First, the tree's growth requires the computation of an error function (usually the sum of squared residuals). This requires all data to be present in memory. For most chemistry-related tasks with annotated training labels, this is manageable, however, it poses a severe limitation at larger dataset sizes(Avellaneda, 2019).

Additionally, the approach of building decision trees is susceptible to a phenomena known as concept drift (Gama et al., 2014; Lu et al., 2019; Schlimmer & Granger, 1986; Widmer, 1996). That is, a set of rules is grown in a decision tree based on the data features. If the distribution of data features shifts over time, the decision tree becomes stale, and accurate predictions can no longer be made. Finally, in decision trees, most approaches such as random forest and gradient boosting decision trees cannot be updated over time and must instead be re-learnt from scratch.

5.5 Large scale machine learning

Machine learning algorithms struggle with large number of examples. These limitations are due to the reliance on the entire dataset being available in memory. For large datasets, this approach is not feasible. Therefore, alternative approaches relying on online learning (OL) are required (Agarwal et al., 2014).

OL is the approach of learning from a smaller subset of training data in an incremental fashion. The size of the subsets is always smaller than the overall size of the dataset. Thus, the subsets can be readily inserted into memory and training performed and OL can be carried out using the gradient descent algorithm.

5.5.1 Gradient descent

When many data points and dimensions are used when fitting a regression model, it is often impossible to find the minimum value of the cost function analytically. To overcome this limitation, instead of solving analytically, the derivative can be computed using gradient descent. Gradient descent works by iteratively computing the cost function until it reaches a minimum or a computational budget is used(Murphy, 2012). This gradient descent process first randomly fits a model to the data, computes the derivative of the cost function, and then updates the fitted model to minimise the derivative. This process iteratively repeats until a minimum is found with each update being controlled by a learning rate parameter. The simplest form of gradient descent is

known as batch gradient descent (BGD) and is where the entire dataset is used to calculate the derivative of the cost function. However, a fundamental limitation of this approach is that the computing of the gradient can be expensive when there are many data points and dimensions. Furthermore, when using a complex non-convex cost function, gradient descent is susceptible to becoming trapped in local optima (Baldi, 1995). For large datasets, this approach is not feasible. Therefore, alternative approaches relying on OL are required.

OL is the approach of learning from a smaller subset of training data in an incremental fashion. The size of the subsets is always smaller than the overall size of the dataset. Thus, the subsets can be readily inserted into memory and training performed. OL can be carried out using the stochastic gradient descent algorithm (SGD) (Kiefer & Wolfowitz, 1952; Robbins & Monro, 1951).

In SGD, random subsets of the training data are used to compute the derivate. These subsets are known as mini batches. This reduces the computational burden and can facilitate the use of vectorised updates. However, a limitation of using mini-batches is that a balance needs to be found between the size of the mini-batch and the computation required; moreover, as the batch size increases, the likelihood of trapping increases, similar to BGD.

An extreme case of SGD used in the vowpal wabbit (Agarwal et al., 2014; *Vowpal Wabbit*, 2021) library is where the derivative is computed on a single point. This single point can be sampled randomly or observed from a data stream. This approach is beneficial for several reasons: 1) The method allows for fast updates and thus many more iterations to be executed. 2) By using only, a single datapoint, less data needs to be held in memory. 3) The method could overcome local minima due to the noisier updates of the derivative.4) The model can adapt quickly if the data generated drifts from the original distribution. However, particularly in the case of noisy updates, this can also be a limitation as it can require a significant number of points before convergence occurs. In this work, vowpal wabbit is utilised, which uses OL. This library was chosen due to extensive optimisations of usage with big data.

5.5.2 Adaptive gradient descent (AdaGrad) and feature normalisation

When using stochastic gradient descent, the learning rate is often set as a constant value. This means that the size of the step taken during training is constant. However, a constant learning rate can lead to several issues. The first issue is that the learning rate assumes no information about the data being learnt. For example, if a molecule is represented using structural keys, then it is assumed that all structural keys appear at approximately the same frequency. In practice, some structural features are very rare in molecules. Therefore, it is helpful to adapt the learning rate to cope with

highly sparse features. This is performed using the adaptive gradient descent (AdaGrad) algorithm (Duchi et al., 2010). AdaGrad changes the learning rate depending upon the data used to train the models. It does this by keeping track of the gradients calculated previously on a feature-by-feature basis. When a new instance of the feature appears, then the learning rate increases a significant amount. This is to prevent overfitting to rarely seen features. In contrast, when a feature appears regularly, the learning rate is low which allows for better fitting on features that occur with a greater frequency.

A significant limitation of the AdaGrad approach is that the summation of gradients is used to calculate the learning rate; as the summation increases, the learning rate decreases to 0. This means that if enough training steps are taken, then the algorithm will no longer improve performance. The most accessible approach to overcome this limitation is through normalisation(Ross et al., 2013).

Normalisation overcomes the assumption that features will be of a similar scale. In practice, this is not true, and therefore normalisation of features is an excellent strategy to improve the performance of stochastic gradient descent. However, as stochastic gradient descent uses a process that means that the entire dataset and thus all features are not read in at once into memory, the normalisation process using OL is non-trivial and relies on storing the magnitude of different features as they are observed. The magnitudes are then used to normalise the features for each data point. This process can be combined with stochastic gradient descent to produce flexible easy to train linear regression models. This process is part of the vowpal wabbit library which is used extensively throughout this work (Agarwal et al., 2014; *Vowpal Wabbit*, 2021).

5.6 Machine learning model validation

Following training of a machine learning algorithm, it is important that a model is tested to access likely performance on unseen data. Testing is where the machine learning model is provided with a set of inputs and predictions are made, and the predictions are then compared to real measured values. For example, in chemistry a model of biological activity of a small molecule against a single target protein could be trained. To assess the model, molecules with known bioactivity values against the single target are predicted using the trained model, and the resulting bioactivity predictions are compared to the measured values using testing metrics.

One of the most common metrics is to calculate the R^2 of the model against a set of data. The R^2 is the coefficient of determination and measures how much variation can be captured by the predictions of a model. For example, the calculation of R^2 is shown below:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})} \quad (8)$$

Where y_i, is the measured data point, $\hat{y}_i$ is the predicted value and $\bar{y}$ Is the mean value. R^2 is bounded between $-\infty$ and 1. A value of 1 would suggest that the model explains all variations in the dependent variable. A value of 0 would suggest a horizontal model that does not consider any variation of the dependent variable. Finally, negative R^2 would imply a negative relationship between the predicted output and the dependent variable.

Several limitations of R^2 have been identified and reported in the literature (Alexander et al., 2015; Barrett, 1974). Firstly, the R^2 does not consider the residuals or coefficients of the model. Therefore, R^2 does not provide an estimate of error. Secondly, R^2 can be susceptible to outliers, leading to spurious high values of R^2.

Alternative metrics are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). These are measured values of the difference between predicted and true values. For example, low values of RMSE and MAE would suggest that the prediction is close to the true value. The equations for computing the RMSE and MAE are shown below:

$$RMSE = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \qquad MAE = \frac{1}{n} * \sum_{i=1}^{n} |y_i - \hat{y}_i| \quad (9)$$

Using the same definitions as described for the calculation of R^2. Here, MAE and RMSE differ in how the difference between true and predicted values are scaled. RMSE utilises a square root rather than an absolute value in the case of MAE. In both cases, the MAE and RMSE capture the model's error.

A core limitation of the metrics described is that they assume that the test data distribution will match the distribution of data that will be seen when predictions are made. As highlighted previously, this assumption is not necessarily true for all data sources and can lead to concept drift. For example, the model may perform poorly if molecules are being generated that lie far outside the training dataset. This is an obvious limitation encountered in de novo design as search methods are likely to traverse outside the original training dataset.

Assessing a model's performance on unseen labelled testing data allows understanding of how the model will likely perform in practice. Frequently, more complex approaches to training and assessing model performance are performed. More complex approaches are utilised to allow for the identification of optimum model parameters. A common approach is to partition the dataset into a training set and a test set, as shown in Figure 5-2, referred to as train-test. However, frequently,

more complex approaches to training and assessing model performance are performed to allow for the identification of optimum model parameters. For example, cross-validation is an approach where the training data is divided into a series of smaller individual train-test problems. Cross-validation is a valuable technique to avoid overfitting(Hastie et al., 2001). This is because partitioning the data into a training set and testing set could inadvertently lead to overfitting where specific features are only present in one dataset and not the other.

The methodology is often further characterised based on the partitioning of the train-test split and the number of individual folds used. Typical cross validation folds might be fivefold or tenfold. In the case of fivefold, a model is trained using four training folds, and testing of the model is carried out on the remaining fold. The testing will return a single metric or set of metrics for that fold. This process is repeated such that every fold is used exactly once as a set of testing data. Following testing on all folds, the metrics are then averaged. Following, cross-validation a set of parameters is chosen corresponding to a specific model, and the entire training data (all folds) is then used to fit a final model. This final model is then tested against the testing set, and the final computed metrics are determined. The ratio of train and test and the number of folds used during cross-validation depend on the dataset size and the domain for which the dataset is derived but typical splits are 80:20 or 70:30 train:test. 70:30 means that the train test problems consist of 70% for training and 30% for testing.

Different approaches to validation can be carried out and are common in machine learning for chemistry. For example, in biological activity prediction, often temporal validation will be carried out (Bosc et al., 2019; Lenselink et al., 2017). Temporal validation is where the data is annotated with a timestamp. During cross validation, the data is partitioned such that the model is trained on data collected at the earliest time stamps and tested on data with later timestamps. Temporal validation mimics the process of data being collected at the beginning of the drug discovery project to build models which are then used to direct the design-make-test-analyse cycle.

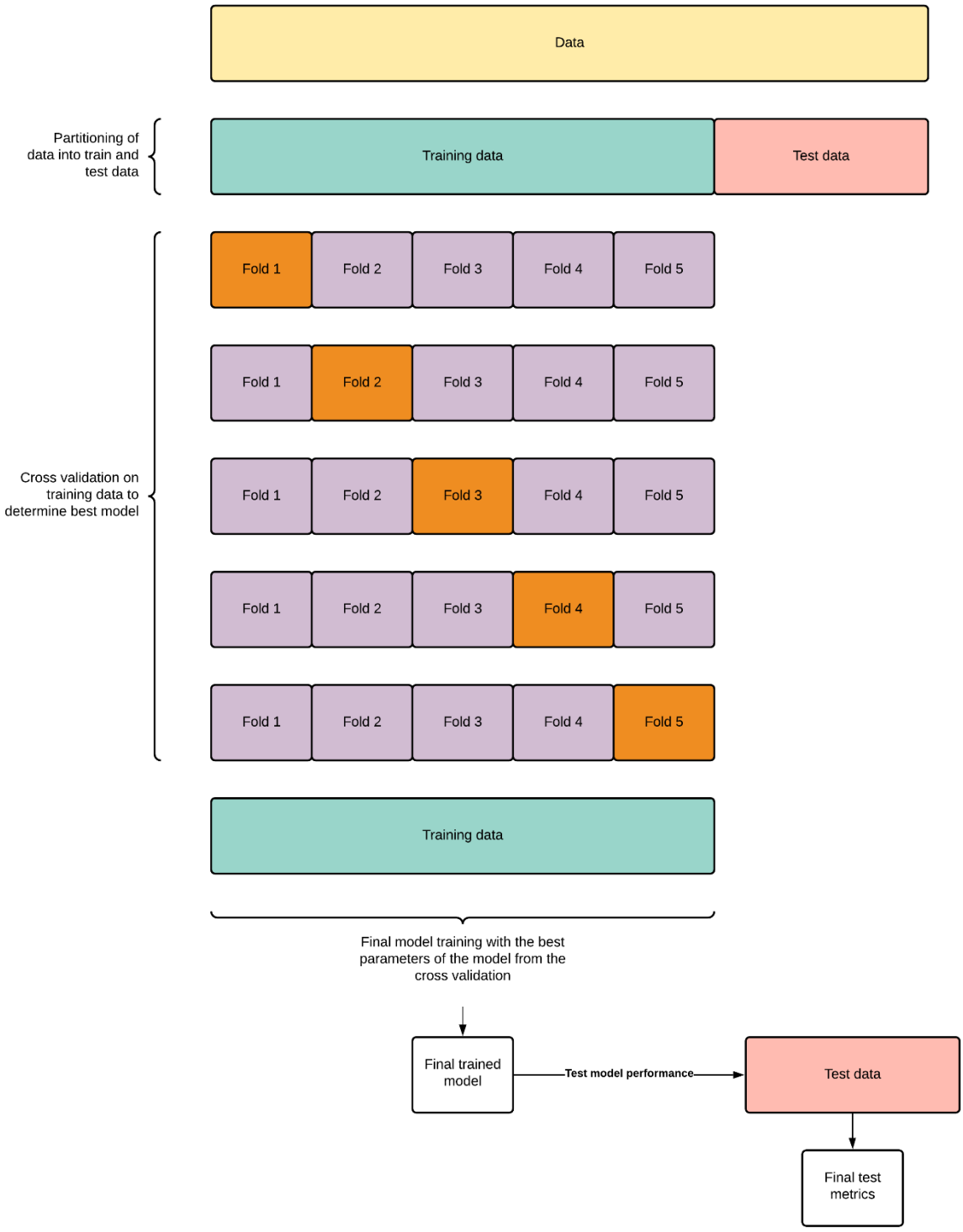


Figure 5-2 Visual representation of the cross-validation process. Adapted from (Scikit-learn, 2021)

5.7 Reinforcement learning

Reinforcement learning is a subfield of machine learning. Reinforcement learning aims to train an agent which interacts with an environment to choose optimal actions(Sutton & Barto, 2018). Several components are required for a reinforcement learning algorithm. An agent that chooses actions and receives rewards. An environment that is an entity that is acted on by the agent and returns a state and a reward. A state is the snapshot of an environment. The reward is a continuous value that the agent acquires when acting upon the environment. The goal of training is for the agent to learn which actions can maximise the sum of rewards that it receives. Figure 5-3 below shows the interplay between agent and environment.

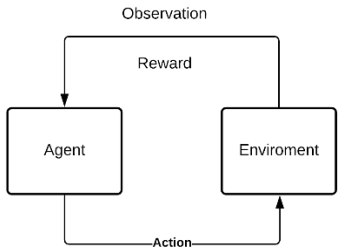


Figure 5-3 Simple reinforcement learning environment

Over successive iterations of the agent choosing actions and receiving rewards, the agent learns the mapping between features and expected reward. In reinforcement learning, the term policy is used instead of model. There are two types of learning: single-step learning and multi-step learning. Single-step learning receives the full reward after a single-step. Multi-step learning receives a partial reward and only sees the sum of partial rewards at the end of the chain of steps. The process of training a reinforcement learning algorithm is described below:

1) Initialise an agent and a starting state
2) Agent chooses an action based on the state
3) The action is applied to the environment
4) The environment returns a new state and a reward to the agent

In single-step learning, steps two to four return a reward, and the agent immediately learns from the reward and the action taken. In the multi-step case, steps two to four are repeated until a terminal criterion is reached. The multi-step agent then learns the mapping between the sum of rewards and the actions taken.

5.7.1 Single-step reinforcement learning algorithm's

The process of learning a mapping between state, action and the reward has been explored extensively. A popular approach to single-step reinforcement learning is the contextual bandit algorithm (CB). The CB algorithm (Bietti et al., 2021; Langford & Zhang, 2009) takes a set of inputs (a context) and predicts a probability distribution. The inputs are the current state the agent is in and the actions the agent can choose from. During training, the CB algorithm is given a state with a set of actions and an action is chosen by sampling the reward which is returned to the CB algorithm. The CB algorithm updates, and the process is repeated until a computational budget is reached. Following training, the CB algorithm should have learnt to build a probability distribution over each action such that actions which yield high rewards are given higher probabilities.

A CB algorithm can be separated into two parts: a learner and an exploration algorithm. The learner learns the relationship between inputs and rewards, and the exploration algorithm maps the relationship between the predicted rewards of the learner and a probability distribution of the rewards. Many different algorithms can be used to learn the relationship between inputs and predicted rewards. A popular approach is to utilise a linear regression model trained via normalised stochastic gradient descent, as described above. The exploration algorithms that make up a CB approach are varied, however, a popular approach is utilising the softmax equation (Cortes, 2018). The softmax approach takes the predicted values and scales them based upon a softmax function. In the case of CBs, the softmax approach is further modified by constructing the probability distribution based on equation (10):

$$e^{\lambda * Score(x,a)} \quad (10)$$

Where λ is a scaling parameter and score is the predicted reward based on the current state x, and action a.

5.7.2 Multi-step reinforcement learning algorithms

In multi-step reinforcement learning algorithms, a partial reward is given at each step and the agent maintains a sum of rewards at each step. During the agent's training, the mapping between an action and the sum of rewards is learnt from each step. As the reward is obtained at the end of a set of steps, this is equivalent to traversing a Markov Decision Process (MDP). An MDP is a generalised extension of a Markov chain described in Chapter 3 and is a directed graph with each state being connected by transition probabilities. Importantly an MDP rewards the agent after each step with a partial reward summed over successive steps.

In contrast to single-step approaches, multi-step algorithms need to map between an action and the final sum of rewards and thus learn the underlying transition probabilities of the MDP. In Chapter 3, a multi-step reinforcement learning (ReLeaSE (Popova et al., 2018)) approach for de novo design was described which utilises the REINFORCE algorithm (Williams, 1992). REINFORCE chooses an action by sampling a deep neural network. The partial reward is then computed via Monte Carlo sampling. The concept of returning a reward at the end of a series of actions forms the basis of the Monte Carlo Tree Search (Coulom, 2006; Kocsis et al., 2006), which is described further in Chapter 6.

5.8 Dimensionality reduction techniques

Frequently chemical datasets are constructed of many high dimensional features. For example, the Morgan fingerprint which is a popular fingerprint choice for machine learning often exceeds 1024 dimensions in length. A limitation of many machine algorithms is the "curse of dimensionality". The curse of dimensionality is where the distances between data points increases as the number of dimensions' increases (Bellman, 1961; Hastie et al., 2001). This leads to models being able to fit to data better due to easier portioning of the feature space, however, the model is likely to be overfit to a sparse distribution of points.

A common approach to overcoming the curse of dimensionality can be achieved through reduced representations (Van Der Maaten et al., 2009) . Reduced representations are the result of applying a dimensionality reduction technique to a set of data to reduce the number of dimensions and fit a better model. Generally, dimensionality reduction techniques rely on methods such as principal component analysis (PCA), t-distributed stochastic nearest embedding (TSNE), or universal manifold approximation projection (UMAP) (McInnes et al., 2018; Pearson, 1901; Van Der Maaten & Hinton, 2008). PCA relies on the correlation between features to reduce the overall number of features that describe a single data point. TSNE relies upon finding a model which minimises the similarity between points. TSNE is t-distributed due to relying upon a T distribution to measure similarity between neighbours. TSNE is stochastic due to relying on a random positioning of points in a lower dimensional (embedding) space than the starting feature space. UMAP relies upon building a series of connected graphs between points in the high dimensional space and then projecting that graph onto a manifold (an object that maintains some distance between points). The manifold is manipulated to minimise the distance between nodes on a connected graph and maximise the distance between nodes in adjacent graphs.

Dimensionality reduction techniques often suffer limitations especially for large datasets. For example, PCA suffers memory and speed limitations when there are large number of data points to calculate the correlations between, and likewise TSNE struggles to minimise the positions of large

numbers of data points (Van Der Maaten & Hinton, 2008; Zhang & Yang, 2018). Finally, UMAP can be limited for large datasets by the requirement of calculating nearest neighbours for a large number of points (McInnes et al., 2018).

5.9 Conclusions

This chapter has provided an overview of the machine learning methods that are relevant to the experimental parts of this thesis including a general overview of the process of training and testing machine learning algorithms. Machine learning algorithms are a popular approach to learning patterns from data. Several commonly used machine learning algorithms were presented including linear regression and decision trees and the memory limitations of both methods were highlighted. OL approaches that overcome memory issues were then presented including stochastic gradient descent and the online version of gradient descent. Moreover, improvements to online algorithms were also provided. Additionally, validation methods of machine learning algorithms were discussed such as cross-validation. Reinforcement learning was then described, including the training process of reinforcement learning algorithms, and different single and multi-step reinforcement learning algorithms. In the final part of this chapter dimensionality reduction techniques were presented such as PCA, TSNE, and UMAP. The next chapter introduces the Monte Carlo Tree Search algorithm, a popular multi-step reinforcement learning approach, and describes how it has been adapted for reaction-based de novo design.

Chapter 6 Monte Carlo Tree Search

Reaction-based de novo design (RBDD) has two main shortcomings. First, RBDD suffers from the "intermediate problem" whereby intermediate molecules in a synthesis path may score poorly against a set of objectives even though they may lead to desired products. Second, RBDD is limited by the combinatorial explosion of molecules generated during the search process. In this chapter, a methodology designed around solving both issues are outlined. An approach known as Monte Carlo Tree Search (MCTS) is used. The first part of the chapter focuses on an overview of the MCTS algorithm leading to a discussion of strengths and weaknesses of the approach. The second section describes the prerequisites needed to run the algorithm and an implementation of the MCTS for RBDD.

6.1 The Monte Carlo Tree Search

The MCTS algorithm (Coulom, 2006; Kocsis & Szepesvári, 2006) explores a search space by growing a tree through the search space to identify nodes that satisfy a goal objective. The MCTS was introduced as an extension of Monte Carlo planning in general game playing. The MCTS explores the search space through a series of node expansions and node evaluations, advancing through the search space until either a computational limit has been reached or the goal has been found. The MCTS approach was designed with the purpose of overcoming massive combinatorial explosions found in domains with large numbers of decisions to choose from, by balancing exploration and exploitation of regions of the search space. This balancing of exploration and exploitation leads to an asymmetric tree being grown.

The MCTS algorithm relies on a tree data structure to explore the search space. Tree data structures are common in computer science and are hierarchical. For example, a simple tree data structure can represent a sequential decision chain. Each decision in the chain is an edge, and a node represents the outcome of a decision. The edges in the MCTS are therefore directed so that the MCTS is a directed-tree data structure. Figure 6-1 shows a simple directed tree structure where the node are coloured, depending on their properties in the tree. The root node is coloured green, and the child nodes are coloured blue. A root node is the starting point for the tree. A child node is generated by applying an action (decision) to a node and is linked to the originating node (the parent) via a directed edge. In Figure 6-1, the green root node is also a parent node connecting to two child nodes, resulting from two different actions. In many tree data structures, each node represents a problem state, and an edge represents the transformation from one state to another.

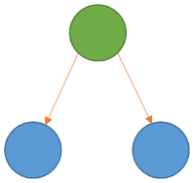


Figure 6-1 Tree structure blue nodes connected by orange edges.

Tree structures are commonly used to represent sequential domains, for example, in games such as Go and chess which are both sequential decision processes. In both Go and chess, a node represents a board state and an edge represents a move which generates a new board state. At each stage in both Go and chess, there are typically many moves to choose from. A player will choose a move on their turn to build a sequential chain of moves from the starting board to the end of the game. The sequential chain of edges and nodes is called a path. The path of moves built up in a game is a single path amongst a large search space of possible paths. The purpose of a search algorithm is to find the best path of moves for the player to win. An approach such as an exhaustive search would involve applying all moves to all nodes from which the best path could be extracted, however, this is unfeasible in most cases. The MCTS was utilised in the game of Go, whereby "Superhuman" performance was achieved (Schrittwieser et al., 2020; Silver et al., 2017; Silver et al., 2016). In the MCTS, a selection function is used to determine which nodes to expand next. The result is that an asymmetric tree is grown, which represents different paths through the search space.

The MCTS algorithm consists of four steps: selection, expansion, simulation, and backpropagation. Selection is the process of choosing a node to expand next to grow the tree. Expansion is the process of generating a new node (or nodes) from the selected node. Simulation is the process of assessing the value of the new node and is achieved by growing a temporary subtree beneath the expanded node and scoring the leaves of the temporary subtree against a goal objective. The scores obtained from the temporary subtree leaf nodes are aggregated and attributed to the expanded leaf node. Backpropagation then takes the score acquired during the simulation stage and adds it to all parent nodes from the expanded node to the root node. At the end of backpropagation, another cycle of four steps begins starting with selection. A single sequence of the four steps is called an iteration. The MCTS continues until either a set number of iterations is complete, a computational budget is used up, or the goal is found.

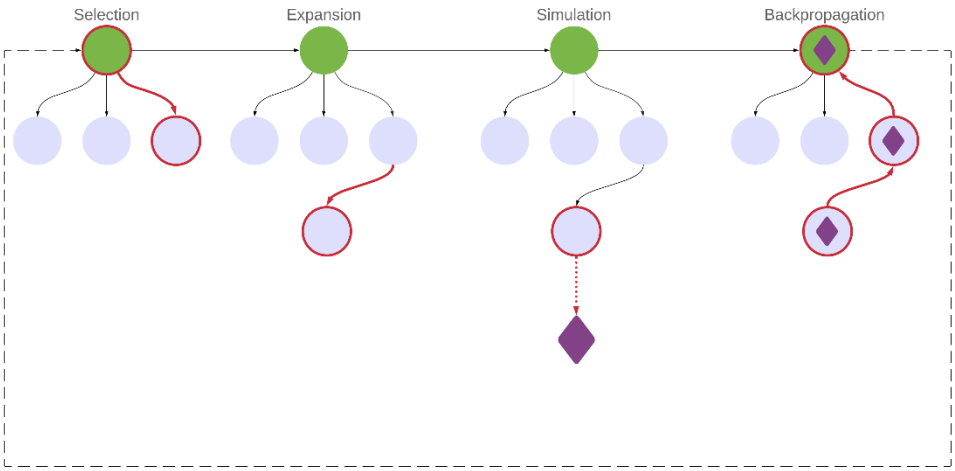


Figure 6-2 Diagram of the MCTS algorithm. Each step is described from left to right, starting from selection, and finishing with backpropagation. After backpropagation, one iteration is completed, and the process starts again from selection with the new expanded node included in the tree with the updated node information.

Figure 6-2 provides an overview of the stages of the MCTS. The MCTS grows through the search space in an asymmetric manner, through each cycle of the algorithm. A detailed description of each step and how the separate steps are combined in the MCTS is provided below.

6.1.1 Basic algorithm

In this section, a description of the basic MCTS algorithm is given. The prerequisite definitions are described first, followed by the node and edge data structures. Then the four steps of selection, expansion, simulation, and backpropagation are described. Where appropriate, examples in domains are provided to illustrate more technical concepts.

6.1.1.1 Prerequisite definitions

The MCTS requires four components to be defined. A state, an action, an evaluation function, and a stopping criterion. Each must be defined before the search occurs. The definitions are described in Table *6-1* below.

Component	***Purpose***	***Example***
State	A snapshot of the search space	Game state, set of items, molecules
Action	A transformation which converts a given state into another	Move in a game, adding or removing items from a set, a RV transformation
Evaluation function	A function that transforms a state to a score that assesses how well the state meets a design goal	A function returning a value, Tanimoto similarity coefficient, QSAR model.
Stopping criterion	The criterion for ending the search	The number of iterations, the goal molecule has been found, or a computational resource has been used up.

Table 6-1 Prerequisites for creating an MCTS including the component, purpose and examples from different domains.

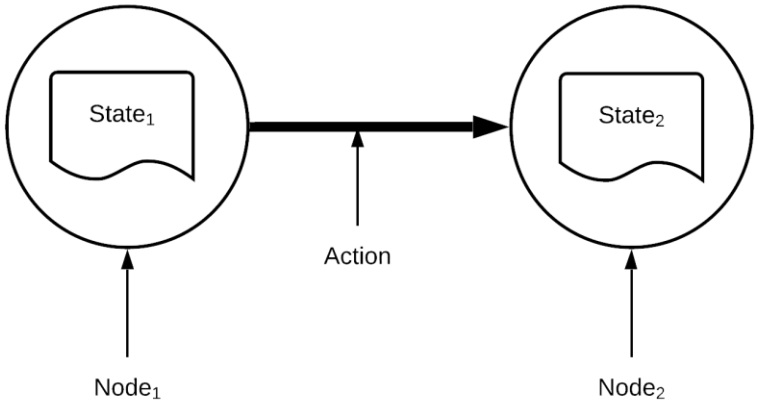


Figure 6-3 A description of a state action pair. The nodes contain the states. The action is a transformation which is applied to state one generating state two. State two is stored in node two.

An example of a state in Go and chess is a specific configuration of the board, that is, the pieces, part way through a game. An example of an action in Go and chess is a single move, which transforms the board from one specific configuration of pieces to another specific configuration of pieces.

The evaluation function determines how states are assessed. The evaluation function takes the state stored inside a node and returns a value that reflects how well the state meets the goal criterion. Depending on the domain, the output of an evaluation may be a single integer or a continuous value. For example, in Go and chess, the evaluation function assesses the winning chances for the player based on the state's specific configuration of pieces on the board.

The stopping criterion defines when the search is terminated. A stopping criterion can either be domain-independent or domain-dependent. For example, the search could end when the number of iterations has reached a predefined limit or met a convergence criterion. A convergence criterion could be (maximal) delta improvement in the objective function score over a set number of iterations similar to the process of early stopping in evolutionary computing (Michalewicz, 1996). Alternatively, the domain itself can dictate the stopping criterion. For example, in Go or chess, a stopping criterion for the search would be when the game has reached a terminal state such as checkmate where a player has won.

6.1.1.2 Node and edge objects

The MCTS builds a tree constructed using node objects. Node objects contain metadata of the states. The metadata is outlined in Table 6-2 below.

Metadata	***Purpose***
Visit count	The cumulative number of times the node and all nodes in the subtree beneath it have been simulated.
Total score	The summation of all scores, obtained through the simulation of the node and simulations of all nodes that exists in the subtree beneath it.
Parent	The parent of the node
Children	The children of the node
State	The problem state the node represents

Table 6-2 A description of metadata stored in a node along with the purpose of the metadata to carry out the search.

In more detail, "visit count" is a cumulative count of the number of simulations that have occurred from both the node and the subtree that exists beneath the node. The "total score" is the summation of the node score returned from the simulation and the scores from the simulations that occur in the subtree beneath the node. Both visit count and total score are updated during backpropagation.

The tree contains three types of nodes: the root node, intermediate tree nodes and leaf nodes.

- Root node: The root node forms the start point for the MCTS and does not have a parent node. The root node is the only node in the tree that is manually defined.
- Intermediate tree node: An intermediate tree node has both a parent node and at least one child node
- Leaf node: A leaf node is a terminal node and does not have any children.

Edges are not defined explicitly in the basic algorithm; they are implicit and represented by storing data on the parent node and child node. More advanced modifications of the MCTS use dedicated edge data structures.

6.1.1.3 Selection

The selection process (shown in Figure 6-5) determines which node is expanded next in the tree. It involves tracing a path from the root node to a leaf node. The order in which the nodes are traversed is determined by what is known as the UCT equation which balances the search strategies of exploitation and exploration. The UCT values are calculated as the tree is traversed as follows:

(1) Given a node (A) -> Calculate the UCT value for all children
(2) Select the child node with the highest UCT value.
(3) Repeat steps one and two until a terminal leaf node is reached.

When the path traced arrives at a terminal leaf node, the selection process terminates, with the terminal leaf node passed to the expansion stage. As a result, a subtree only is traversed during selection and the single leaf node in the subtree with the highest UCT value is chosen.

The UCT equation is defined as follows:

$$UCT = \bar{X}_j + 2C_p\sqrt{\frac{2\,ln\,n}{n_j}} \quad (11)$$

$\bar{X}_j$ is the mean reward (total score divided by the total number of visits) obtained so far; C_p is an exploration parameter used to tune the amount of exploration; n_j is the number of visits to the node so far; and n is the number of visits to the node's parent.

The UCT equation can also be written as:

$$UCT = \frac{Total\ Score}{number\ of\ visits} + 2 \times C_p \times \sqrt{\frac{2\ln(number\ of\ visits\ to\ parent)}{number\ of\ visits}} \quad (12)$$

Where the total score is

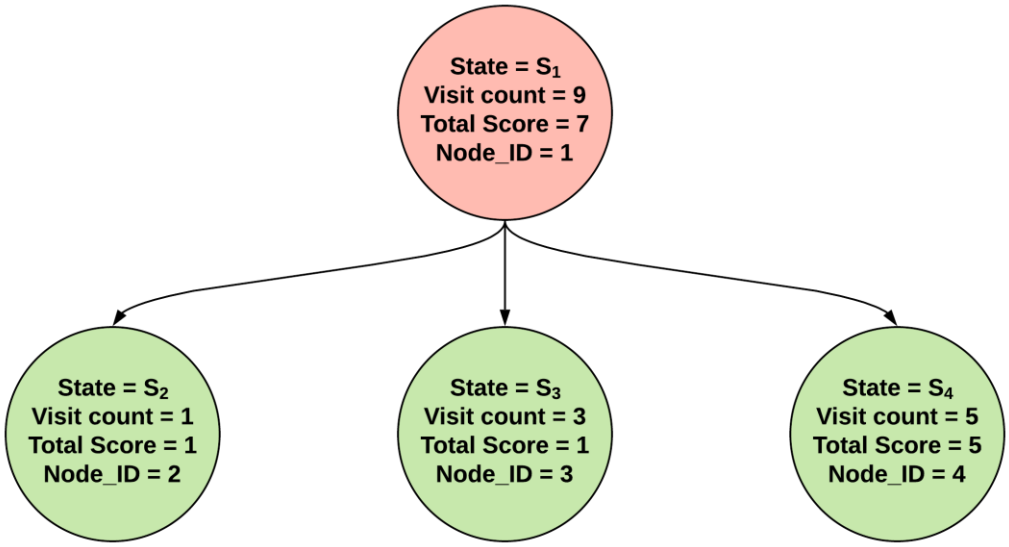
$$Total\ Score = Score\ of\ the\ current\ node + \sum score\ of\ children \quad (13)$$

The two terms in the UCT equation can be separated into separate terms:

$$UCT = Exploitation + Exploration \quad (14)$$

This function is recursive, for example the total score is the score of the current node and sum of the current node's children. The scores of the current node's children are their scores and the summation of their children's scores and so on.

A brief worked example of the UCT value being calculated for a node with three child nodes which are also leaf nodes is shown in Figure 6-4.



The UCT scores for nodes two, three and four are shown below using equation (12) :

$$UCT2 = \frac{1}{1} + 2 \times C_p \times \sqrt{\frac{2\ln(9)}{1}} = \mathbf{1 + 4.19C_p}$$

$$UCT3 = \frac{1}{3} + 2 \times C_p \times \sqrt{\frac{2\ln(9)}{3}} = \mathbf{\frac{1}{3} + 2.42C_p}$$

$$UCT4 = \frac{5}{5} + 2 \times C_p \times \sqrt{\frac{2\ln(9)}{5}} = \mathbf{1 + 1.87C_p}$$

Figure 6-4 A worked example of the selection process is shown. The process occurs every time the selection routine is running.

For a given node, the UCT is calculated for each child node and the one with the highest value is traversed next. The example highlights two practical considerations relating to the usage of the C_p term. If C_p is high, then selection of node S2 occurs even though node four has a higher total score, and this is, therefore, an example of exploration with node four penalised relative to node two due to it having been visited more. For $C_p > 0$ then the order of the UCT scores is UCT2>UCT3>UCT4. In

contrast, if $C_p = 0$ then node two and node four have equal UCT scores. If two or more nodes have the same UCT value, then a tie occurs. Depending on the domain, ties can be common or infrequent. The standard approach is to break the tie at random. In the example above, if node two is a leaf node, and if $C_p > 0$, then selection terminates here, if not then the process repeats for the children of node two until a leaf node has been reached.

Several behaviours arise from the selection process:

- All the data used to compute the UCT equation is stored in the metadata in the nodes, therefore, the selection stage is very fast to compute.
- C_p dictates how often less-visited nodes are seen. Higher values of C_p lead to more exploration and a broad tree with nodes with poor scores received during the simulation phase being visited more often, compared to lower values of C_p. This is because for high values of C_p the second term in the UCT dominates so there is little to distinguish between nodes.
- A small value of C_p leads to a deep tree as highlighted above when the C_p term is set to zero.
- For child leaf nodes with a visit of one, the node with the highest total score is always selected so that the selection of leaf nodes is always greedy.

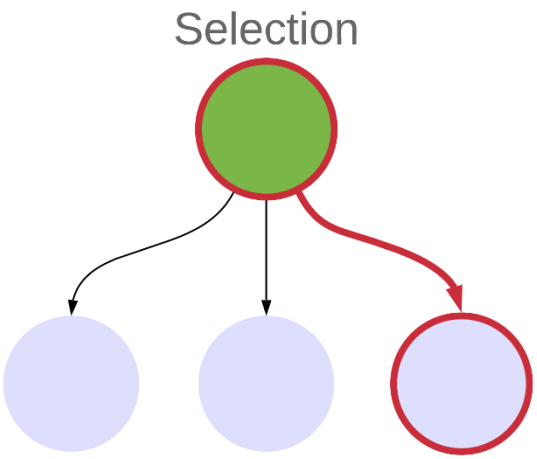


Figure 6-5 Diagram showing the selection process moving down the tree

The selection routine is vital in minimising the combinatorial explosion in two ways. First, selection returns a single node to pass on to the expansion stage and leads to throttling of the growth of the tree. Second, selection explores a single subtree of the total tree leading to certain parts of the tree never being explored.

6.1.1.4 Expansion

Expansion (shown in Figure 6-6) is the process of generating new states which are stored in nodes. Expansion involves the application of an action. Depending on the domain, an action may generate one or multiple states. The states are then stored in one or multiple nodes, respectively. Expansion is domain-specific and, for example, in Go and chess, one board move gives one state.

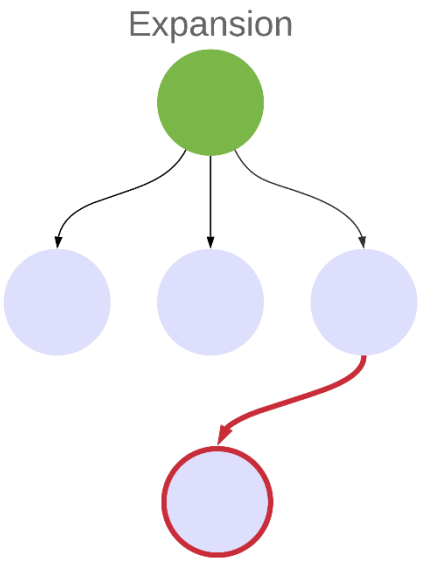


Figure 6-6 Shows expansion being applied, and new child (state) being created

The expansion process can be improved by the exploitation of domain-dependent characteristics or action prioritisation. These improved expansion methods can be based on machine learning approaches or nuances of the domain.

6.1.1.5 Simulation

The simulation process (shown in Figure 6-8) is the evaluation step of the MCTS. A temporary subtree is grown beneath the node generated during the expansion phase and the evaluation occurs at the states contained in the subtree. The growth of the subtree relies on the same process used in the expansion phase, whereby an action is applied to a state and generates a new state. The subtree grows until a stopping criterion is reached, the stopping criterion can be domain-independent or domain-dependent. Domain independent criteria can be features such as the depth of the subtree from the expanded node to the terminal leaf node. Domain dependent stopping criteria can be the game-ending, or a particular feature of the state being achieved. When the subtree is terminated, the evaluation function is used to score the states found in the terminal leaf nodes.

The simulation routine occurs as follows.

1) Given a state, apply a single valid action chosen at random.
2) Generate a new state.
3) Repeat steps one and two using the new generated state each time in a sequential manner until the stopping criterion is met.
4) Evaluate the final state at the end of the subtree.

The score generated at the end of step four of the simulation routine is the score which will be used by the backpropagation phase. The evaluation function can generate a score which is either continuous or discrete depending on the problem domain. Only the terminal state of the subtree is evaluated. By evaluating the end state, the simulation routine allows for the passing over of local minima and intermediate states and aids in solving the "intermediate problem".

Modifications of the simulation subroutine are common and are carried out to maximise the amount of information gained while concurrently minimising the run time of the simulation. In the description above, the simulation routine occurs in a sequential process with each state being used to generate new states in a single chain until the final state is reached. The simulation subroutine can be modified to have multiple actions applied in parallel generating multiple chains forming a directed tree from the starting state. If multiple terminal states are generated, then mathematical transformations can be introduced in the simulation routine to transform the end states into a single value that is attributed to the expanded node. The process of transforming multiple evaluations into a single value can lead to very accurate evaluations of the expanded state's performance. For example, a simulation routine that generates a single chain result in the evaluation of a single endpoint. Whereas a simulation routine which allows for 1000 chains to be generated in parallel will lead to 1000 terminal states being reached and evaluated with the scores being transformed into, for example, a mean. The parallel routine estimates the average value of the expanded state much better than the evaluation at the end of the single routine. As the evaluation function is called at the end of a simulated sequence, if the simulation ends on a poor scoring state, the expanded node will also score poorly. In contrast, a simulation with multiple actions averages out the performance, and is therefore more likely to lead to a better score being assigned to the expanded node.

There are different ways of incrementing the visit count for the expanded node when multiple endpoints are generated during simulation. The visit count can be incremented once for all endpoints or once for each endpoint. If the visit count increases once for each endpoint, then the phrase "multiple simulations" is used.

In domains where the application of an action is computationally expensive, the simulation routine can be challenging to execute computationally. In contrast, domains where the application of an action is cheap, allow for a large subtree to be generated in parallel with many terminal leaf nodes. Since the simulation is applied to each newly expanded node in the MCTS, the MCTS algorithm scales poorly with the number of nodes that are generated. Figure 6-7 demonstrates the increasing computational expense unfolding from a range of simulation times.

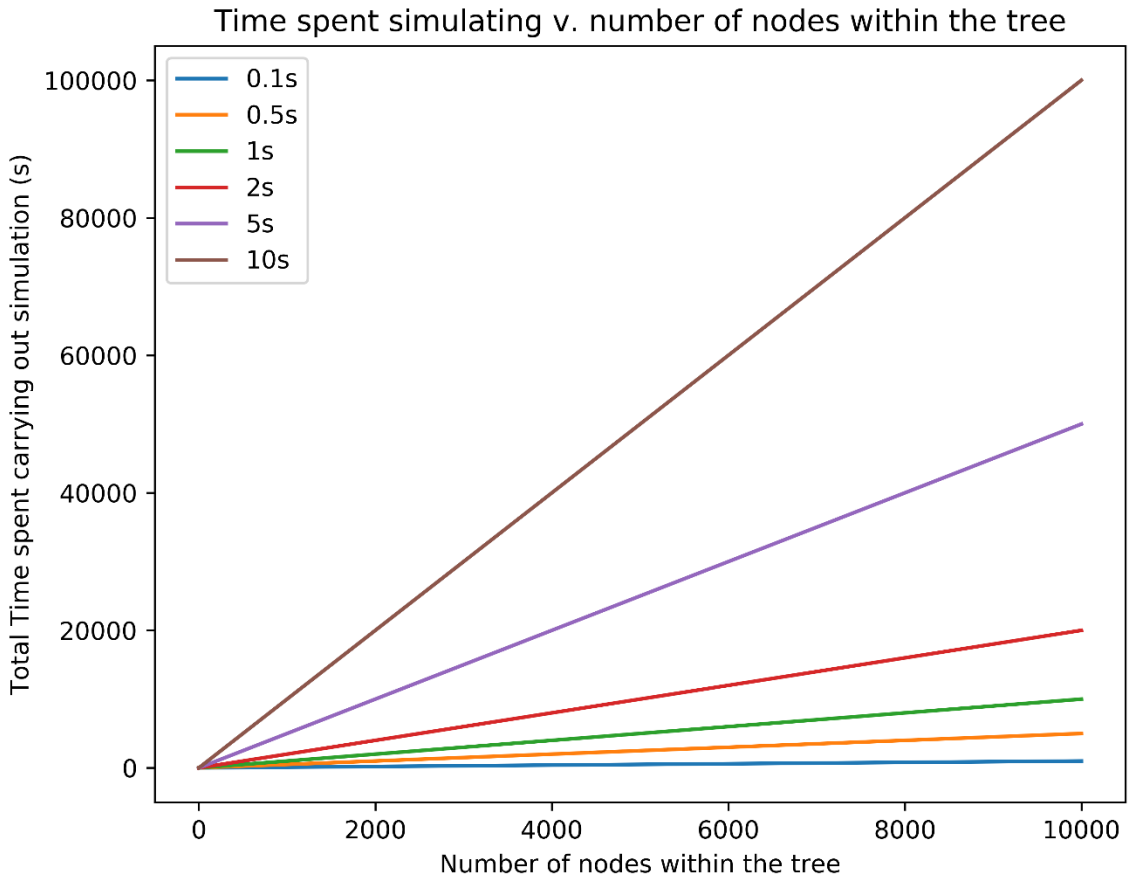


Figure 6-7 Demonstrating the effect of increasing the time spent in simulation versus the total number of nodes in the tree. Each line represents a separate hypothetical search with simulation taking different lengths of time. The brown line is a search where each simulation step takes 10s, in contrast the blue line represents a search which where each simulation step takes 0.1s. The effect of having a longer simulation time leads to a "knock on effect" in terms of computational time as the tree grows.

In practice, there is a trade-off between accuracy and speed in the simulation and across all domains, a major aim during implementation is to reduce the time taken when calling the simulation routine.

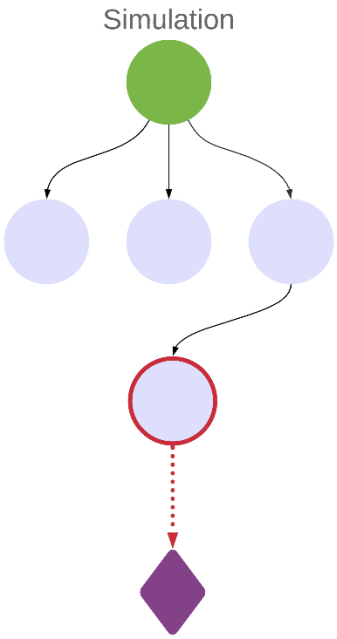


Figure 6-8 Shows the simulation stage extending to a terminal state

6.1.1.6 Backpropagation

The final stage of the MCTS is the backpropagation step (shown in Figure 6-9). The backpropagation step reinforces the path identified during the selection process with the new information generated during the simulation phase. Backpropagation starts at the expanded leaf node with the new score generated during the simulation phase being added to the score of the expanded node. The score is then added to each parent node's metadata in turn, following the same path traced during the selection phase. The backpropagation routine occurs as follows.

1) Add the score returned during the simulation routine to the expanded node to form the total score used in the UCT equation.
2) Increase the visit count of the expanded node by 1. This is the visit count used in the UCT equation
3) Identify the parent node
4) Increase the visit count of the parent node by 1.
5) Add the score returned during the simulation routine to the total score of the parent node.
6) Repeat steps 3-5 moving up the tree until the root node has been updated.

Backpropagation updates the metadata of the nodes. Updating the path traced during selection from the newly expanded node to the root allows the path to be reinforced or weakened with the updated information. Backpropagation allows previously poor scoring paths to be updated and reinforced with positive information obtained from the simulation routine. In contrast,

backpropagation also allows for previously good paths that reach a dead end to be weakened. As backpropagation updates visit count and total score once per visit, strong performing paths take many iterations to weaken, and weak performing paths take many iterations to improve.

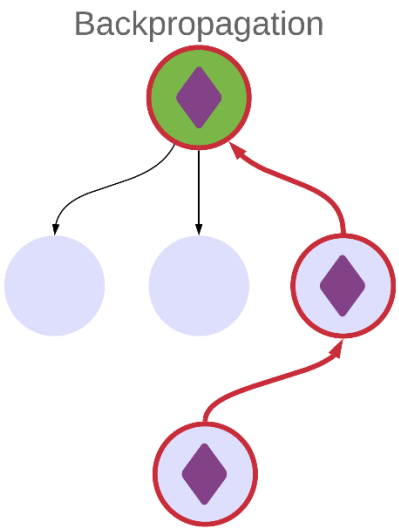


Figure 6-9 Backpropagation of terminal state information up the tree

6.1.1.7 Policies

A policy is a strategy used to direct the search. In the MCTS two policies exist: a tree policy, and the default policy (shown in Table 6-3). The tree policy controls the growth of the tree. In the MCTS, the selection and expansion phases are controlled by the tree policy. The default policy controls how the simulation routine is constructed. Backpropagation updates information in the tree and is not part of either the tree or the default policies.

Policy	Purpose
Tree	Direct the growth of the tree
Default	Direct the simulation and evaluation of the nodes

Table 6-3 Purpose of the Tree policy and Default Policy

In a standard implementation of the MCTS with no modifications, the tree policy is the UCT equation with a single application of an action during the expansion phase. The default policy determines how the simulation routine is implemented. In a basic implementation, the default policy consists of a single simulation.

6.1.2 Summary

This section provided an overview of the basic algorithm and each subroutine used in the MCTS approach. A complete iteration of the approach is independent of the previous one. The values of the metadata in each node updated during backpropagation are known at the beginning of the next iteration, all other values used during the search are calculated from the metadata. As the MCTS is a multi-step algorithm, a full worked example including calculations of the UCT equation at each stage is provided in appendix A.

The next subsections describe the strengths of weaknesses of the MCTS.

6.1.3 Strengths of the MCTS

The section discusses a number of strengths of the MCTS algorithm. First is that the algorithm can manage the combinatorial explosion while simultaneously handling local minima. The combinatorial explosion is mitigated by the selection phase, while both the simulation and backpropagation routines handle the local minima issue. All parts of the MCTS work in tandem while allowing the MCTS to converge on a selected path. The MCTS will converge upon the best path given a high enough exploration constant and enough computational time (Browne et al., 2012). However, guarantee of convergence to the best possible action only applies when a standard implementation of the UCT is used with no modifications and assumes an unlimited computational budget. The speed at which the convergence occurs depends on the average number of child nodes generated per state, which is known as the branching factor and on the domain being explored. When convergence occurs the same node will always be selected and over several iterations of the MCTS algorithm an asymmetric tree is generated. By choosing the best node, this means certain portions of the search space go unexplored. The asymmetric nature of the search tree prevents the combinatorial explosion of products generated. This is of particular benefit in search spaces with a high branching factor such as reaction based de novo design.

The MCTS is an iterative approach to searching; hence the search can stop at any time (even in an iteration). Not only does this allow the MCTS to be paused, but it also allows the growing of large trees to be saved and distributed across computational resources. Thus the MCTS can be interrogated at each step which ultimately allows the MCTS to be an easily interpreted transparent algorithm.

The MCTS algorithm approaches the problem of traversing a search space via forward sampling. Forward sampling is how humans would traverse an unknown search space. Humans often look a few steps into the future to ascertain the value of making a decision. Forward sampling is used as it prevents being trapped in local optima and mitigates the “horizon effect”. The horizon effect is that

there could be a move (n+1) steps away, causing a significant change to the value of the decision being made at step n, yet this move is not seen as it is beyond the horizon of an algotihm. Forward sampling mitigates the horizon effect by looking beyond the immediate horizon. The horizon effect is the same as the intermediate problem in RV-based search where intermediate molecules generated along the sequence of RVs can score poorly compared to the final molecule in the sequence.

6.1.4 Weaknesses of the MCTS

While there are a multitude of benefits in utilising the MCTS approach as outlined in 2.1.10, there are several key limitations which need to be discussed. These weaknesses are frequently overcome with the appropriate modifications.

When the search space being explored is highly branched and each node has large numbers of possible children to choose from, a basic implementation of the MCTS can struggle to achieve convergence to the correct solution in an appropriate time frame. The reason why substantial branching factors can be an issue is that it is not computationally tractable to simulate such a large number of children.

Furthermore, a problem can arise when a decision between two similar actions must be taken. For example, when individual UCT scores are very close in value, and their difference does not manifest itself until deep in the search tree, it can require a lot of sampling and the construction of a large tree to separate the values of these individual nodes. Hence, the amount of computational resources to separate these decisions can be significant.

A further limitation with introducing a large number of MCTS modifications is that they can lead to a severe computational performance decrease and can lead to the node score value being poorly estimated. As a result, this may lead to the strongest performing decision sequences not being found (Berthier et al., 2010). Enhancements can provide both positive and negative benefits when used together.

Comparing enhancements to the basic implementation is therefore essential to understand if improvements are occurring. From the domain explored, there should be a metric to define success. Often, the primary metric used is the target objective; however, there are many secondary measures that can be used to evaluate success, such as features of the tree grown.

6.2 MCTS Analysis

This section describes different factors which can be used to analyse a tree search. These are separated into three sections: Algorithm performance, Resource usage, Robustness (Barr et al., 1995).

6.2.1 Algorithm performance

The most important aspect of algorithm design and development is performance against the problem being solved. In the MCTS, performance will depend upon the domain being explored. In games, it can be the highest score, or the win percentage against an opponent. In de novo design, it could be the molecule which has the maximum score of the molecule found during the search such as similarity to a known active compound or a QSAR score. Further extensions of algorithm performance can focus on average score of the states found. In all cases algorithm performance is often compared to a baseline, so performance can be measured relative to another method.

6.2.2 Resource usage

When an algorithm is implemented, an important metric to understand the performance of the algorithm is the amount of resources used. An important resource in the MCTS is the computational time used for the search to complete. In particular, the MCTS as a reinforcement learning algorithm, benefits, from more computational time but resources are limited. Therefore, understanding of the MCTS resource usage such as the amount of time take for the search to complete, how many states are generated, and branching factor is important in understanding the overall computational efficiency of the approach.

6.2.3 Robustness

When an algorithm is developed, it is important to understand how the algorithm performs over multiple different problem types and instances. The MCTS was originally designed to be used in many different domains and robustness would be tested by applying the same implementation of the MCTS to multiple domains and comparing the same implementation to the baseline in each domain respectively. This would allow a qualitative assessment of algorithm robustness.

The above metrics are important for understanding the overall performance of an algorithm approach. The metrics, however, need to be combined with appropriate experimental design to understand specific algorithm performance. For example, the metrics shown above can be used to investigate performance of a modification to the base MCTS algorithm. This is the basis of many domain specific applications (Browne et al., 2012).

6.3 MCTS in Chemistry

The application of the MCTS in the chemistry domain has only occurred recently and has focussed on retrosynthesis and atom-based de novo design. First, retrosynthesis is the process of breaking down a molecule in a stepwise process to identify synthetic transformations which can be used to build the molecule up from SMs. Second, the MCTS was used in atom-based de novo design to construct sequences of SMILES characters which correspond to molecules satisfying predefined objectives. Both retrosynthesis and atom-based de novo design suffer from a combinatorial explosion of decisions at each state hence the use of the MCTS. While both retrosynthesis and atom-based de novo design are examples from the chemistry domain, as the problems are different, this has led to differences in the implementation of the MCTS for each problem.

First, as part of the AlphaChem project (Segler et al., 2018), the MCTS and three separate neural networks were used to obtain high-performance retrosynthetic planning.

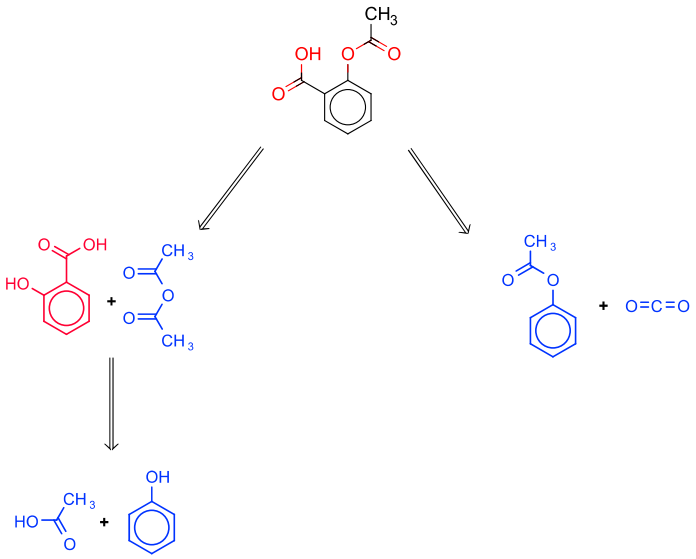


Figure 6-10 Example of a retrosynthesis plan – The blue molecules are reagents which have been identified in a database. In contrast, the red molecule is a molecule that needs to be further split apart. The starting molecule at the top is to be synthesized(aspirin), the red molecule is an intermediate. The blue molecules exist in the database.

Retrosynthetic planning is illustrated in Figure 6-10 which shows a simple tree unfolding from the molecule aspirin when planning a synthetic scheme. Synthetic planning is computationally expensive to solve due to the large number of actions to explore for each intermediate molecule that needs to be split apart. The actions in AlphaChem are molecular transformation rules mined from a large corpus of reaction information consisting of 12.4 million single-step reactions, which were cleaned and separated into two sets — one for expansion and one for simulation. The expansion set contains

more robust reactions compared to the simulation set. The reasoning for this is the simulations are not added to the tree and instead are only used for node evaluation, so transformations being valid is not a priority, instead what matters is that the transformations can be performed quickly and efficiently. The expansion set is used to grow the tree where a higher success rate of the number of valid transforms is essential.

In AlphaChem, three deep neural networks were constructed from the simulation and expansion transformation sets, respectively. The neural networks used in expansion and simulation have two distinct functions in the retrosynthesis planner. First, in the expansion policy network, the network is trained to select a limited number of transforms from the expansion set for a given state. The network acts as a filter to select the most feasible transforms to apply. The second neural network is an "in-scope" classifier which is an additional filter which aims to determine when a reaction transformation is applicable and is trained on a set of scrambled reactions. Given a set of reactions, the products of the reactions are scrambled, and a classifier built whose purpose is to classify the unscrambled reactions as one and the scrambled reactions as 0. As input, the classifier takes the product of the reaction and the reaction fingerprint, as a concatenated input. The underlying assumption is that the scrambled reactions themselves are applicable, while this might not always be explicitly true. The expansion phase takes about 90ms to identify, filter, and apply reaction transformations. The third neural network, which is used for simulations, is less accurate but is faster and takes 10ms to identify applicable transformations from the simulation set compared to 90ms for the expansion process. The training process for the networks occurs offline before the search begins and involves using the corpus of reaction examples to "learn" which reactions are applicable.

In many respects, the usage of two separate sets of actions one for simulation and another for expansion is a modification of the MCTS. The use of two sets of actions of two different sizes, can lead to a possible asymmetry in the search. The search space through which the tree is grown is a subset of the larger space encompassed in the simulation set. When the simulation set is used in conjunction with the simulation routine, the estimations during the simulation reflect the performance of a given node in a different search space. A further modification to the MCTS in AlphaChem is the use of a modified version of the UCT equation, this modified UCT equation, uses the probabilities of a reaction transformation being applicable from the expansion neural network directly in the UCT equation to balance the exploration and exploitation during the search.

AlphaChem demonstrated that solutions could be generated to solve previously difficult retrosynthetic problems with the use of the MCTS and a series of neural networks. By using neural networks in the MCTS, the interpretability as to why a particular reaction is chosen is lost. Moreover,

by using scrambled reactions and a data-driven approach to identify applicable reactions no physicochemical characteristics of individual reaction classes are considered. Finally, by scrambling the reactions this does not guarantee that the reaction could not occur in practice.

Retrosynthetic planning is well suited to MCTS as the algorithm mimics the forward sampling-based behaviour of chemists. Chemists often “look” several disconnections into the future to decide if a disconnection from the current molecule is good. Furthermore, retrosynthesis benefits from a large, well-defined solution space. The solution space is often constructed from large catalogues of commercial building blocks or small molecules. Finally, retrosynthesis can be framed as a more classical game framework. The scoring can be framed as a 0/1 win or loss scoring function where one is the molecule exists in the catalogue and 0 if the molecule does not exist in the catalogue.

Following the successful implementation of the MCTS algorithm for retrosynthesis, attempts have been made to apply the MCTS to de novo design. The ChemTS (Yang et al., 2017) program uses the MCTS to generate SMILES strings which correspond to a molecule with desired characteristics. The MCTS implementation was modified by using the SMILES characters as nodes, and the edges to represent bonds. The selection stage chooses a character in the SMILES sequence to explore next, and 30 new child nodes are expanded at once. Each newly expanded node corresponds to a new SMILES character attached to the sequence. Each SMILES fragment produced in the tree is then passed to a neural network in the simulation phase where the SMILES fragment is completed to a final molecule. The molecule is scored and the score is backpropagated up the tree. Over time this leads to the MCTS driving a global search with the neural network performing a local search via the simulation. An interesting feature of ChemTS is that it only produces valid high scoring molecules in the simulation step, not in the tree itself. The tree, itself contains only SMILES fragments which are then completed during the simulation phase.

ChemTS suffers from the drawback of using SMILES strings to generate the molecules, that is the generative model that is used during the search is likely to generate molecules that are in the training set, the resulting molecules lack structural diversity and synthetic accessibility is not taken into account. Furthermore, time is required to train the neural network and allow for the network to learn the underlying grammar of a SMILES string, although, this only has to occur once. Yang and coworkers do clarify the method can produce 40.89 molecules/min but do not state the percentage of these which are valid upon generation. The approach of ChemTS was further explored, in NMR-TS (J. Zhang et al., 2020),SBMolGen (Ma et al., 2021), and GB-GM-MCTS (Jensen, 2019) which was tested as part of the GuacaMol benchmark (Brown et al., 2019).

It is important to note AlphaChem and ChemTS have been assessed based on algorithm performance and resources used as described in section 6.2. In AlphaChem, this is the number of routes successfully solved and in ChemTS it is the score of the designed molecules. Both methods further discuss resources used based on the time taken to generate solutions. However, the problem domains of AlphaChem and ChemTS are substantially different. AlphaChem is a method for retrosynthesis and is therefore not directly comparable to the method developed in this thesis. ChemTS is a de novo design method and therefore aims to address the same problem as the method developed in this thesis, however is an atom based de novo design method and therefore cannot be easily compared to non-atom based de novo design methods.

Finally, while the MCTS is an obvious choice for domains with high branching factors, several other tree search approaches exist and have been applied in chemoinformatics and de novo design (see section 3.5.2). These include breadth-first search (BFS)(Lee, 1961; Moore, 1959), depth-first search (DFS) (Tarjan, 1972), A*(Hart et al., 1968), Iterative deepening A* (IDA) (Korf, 1985), and α-β minimax search (Brudno, 1963; Knuth & Moore, 1975; Samuel, 1959). In the case of BFS and DFS, these search approaches struggle when handling large branching factors. Both approaches struggle with storing vast numbers of molecules in memory while being sufficiently performant to search through the tree. Alternative search methods are A*, and IDA* which are both heuristic-based search methods. However, both methods struggle in domains where heuristic functions are hard to define. Although these heuristic approaches are not incompatible with the problem of reaction vector-based de novo design, the time required to define an appropriate heuristic function can be excessive. Finally, the problem of de novo design can be defined as "single-player"; therefore, approaches that rely on the minimisation of opponents moves such as minimax with alpha-beta are less appropriate as there is no adversarial element to the problem.

6.4 Methods and implementation

This section describes the implementation of the MCTS algorithm for RBDD which will be referred to as RVMCTS. The section begins with the prerequisite definitions and then describes the selection, expansion, simulation, and backpropagation steps. The application of the MCTS to the problem of RBDD is such that some modifications to the basic implementation of the MCTS are required to adapt the algorithm for RBDD specific features. When the implementation differs from the basic implementation, examples and reasoning are provided.

6.4.1 Reaction Vector Monte Carlo Tree Search

The implementation is written in JAVA with a python interpreter for evaluation, which allows for object-orientated programming structures via JAVA to be used for the MCTS and the flexibility of the Python scientific programming ecosystem. Seven components must be defined to implement the RVMCTS search. The definition of each component used in the RVMCTS is provided in Table 6-4 below.

Component	***Definition***
State	Molecule represented as a SMILES String
Action	The application of a single applicable RV and reagent
Evaluation function	Tanimoto similarity to a pre-defined target molecule using an RDKit Morgan fingerprint 2048 bits in length with a radius of 2
Stopping criterion	50 iterations
Exploration constant	0.001
RV database	H2 Database containing a store of RVs
Reagent pool	A file containing a list of SMILES strings representing reagents

Table 6-4 Definitions of each component in the RVMCTS

In the table above, the definitions allow for minimum memory footprint and fast manipulation where appropriate. The evaluation function is the Tanimoto similarity to a predefined molecule. The choice of Tanimoto similarity using a Morgan Fingerprint of 2048 bits in length with a radius of two was twofold: first Tanimoto similarity is a standard similarity metric used in cheminformatics (see Chapter 2) and second the Tanimoto similarity is fast to compute.

The stopping criterion was set at 50 iterations. The choice of 50 iterations was primarily based on the expected runtime of a structure generation of a single molecule using a reaction vector database and a modest reagent pool. For example, Ghiandoni (Ghiandoni et al., 2020) demonstrated that a single expansion of one molecule would take approximately 38 seconds utilising a RV database of 11.6k. In the case of the JMedChem database, fewer reaction vectors are being used compared with Ghiandoni (11.6k vs 4.5k) hence, it would be expected that each enumeration takes approximately 15 seconds to complete. Additionally, each molecule generated during the expansion is then simulated, utilising the enumeration. Moreover, by sequentially growing a tree, the time expected for each iteration to continue will increase as the search progresses based upon the expected

branching factor. Unfortunately, prior to the search, it is unknown what the branching factor will be. Hence 50 iterations were chosen as the stopping criterion as a default choice. Different evaluation functions and different values for the stopping criterion are explored later in the thesis. The values chosen for these experiments are to understand the algorithm's performance on a problem of restricted scope.

The exploration constant is a hyperparameter that has been explored extensively in the MCTS literature (Browne et al., 2012). The exploration constant will take any real value and is usually bounded between 0->∞. In practice, this value is tuned per problem. The choice of 0.001 for these experiments is based on the expected Tanimoto similarity of a set of randomly sampled molecules. This average value was determined to range between 0.3 and 0.4 for a binary fingerprint (Godden et al., 2000; Todeschini et al., 2012). It is therefore assumed that the average similarity of the molecules generated in a subtree of the MCTS will be in the same range. In this case, the exploration factor should be set to a value that forces the exploration term to be within the same order of magnitude. It is important to note that the exploration constant is multiplied by the exploration term as shown in equation (12). Finally, it is known that the exploration term, when tuned high, leads to more exploration and thus a broader tree. In the rediscovery problem setting, a deeper tree needs to be grown. As a result, there needs to be a slight bias towards exploitation. However, in advance of the search occurring, it is unknown what the optimal exploration parameter value should be. Hence, the value of 0.001 was chosen as a starting point to demonstrate the algorithm's performance but not necessarily tuned to the most optimal choice. Ideally, for each ablation study, an exhaustive grid search would be carried out to find the most optimal choice of exploration parameter for that study. However, this would be too computationally expensive to carry out based upon a similar analysis of the number of iterations.

6.4.2 Nodes and Actions (molecules and reactions)

In the RVMCTS a problem state encoded within a node is a single molecule. The representation used is a SMILES string of the molecule. The SMILES representation was chosen as SMILES has a small memory footprint and interfaces well with the RV engine (Wallace, 2016). A node also stores metadata associated with the RVMCTS. No modifications to metadata are made from the basic approach outlined in Table 6-2.

Only the root node at the beginning of the search needs to be defined. This root node contains the starting state for the search and is created by providing a start SMILES structure from which the tree grows.

6.4.3 Selection - Choosing which node to investigate

Selection proceeds as outlined above and is based on the UCT equation. No additional modifications are made.

6.4.4 Expansion - Taking a node and applying an action

Expansion is a domain specific process. The main difference from the basic MCTS implementation is that rather than a single action being applied, all applicable actions (RVs and reagents) are applied at once. Following selection, the expansion routine is called as follows

1) For the molecule contained in the selected node, query the RV database for all applicable RVs.
2) For each applicable RV determine all applicable reagents
3) Apply all unique combinations of RVs and reagents to the state contained in the selected node
4) For each new product generated, form a new node referencing the selected node as the parent node
5) Pass all newly generated nodes to the simulation stage.

The RVMCTS expansion process differs from the basic implementation of the MCTS to take advantage of the RV engine. The expansion routine in the RVMCTS is an example of leaf parallelisation (Chaslot et al., 2008) and is carried out to avoid querying the database too many times. Furthermore, querying the database and reagent pool and then generating all possible structures in one pass saves on the computational expense compared to repeated calls to the expansion step with a single action per expansion. Applying multiple actions in each expansion also speeds up the growth of the tree.

6.4.5 Simulation - Ascertaining molecule score

The simulation takes each newly expanded node in turn and seeks to assess the performance of the node by growing a subtree from the node. The RVMCTS-simulation sub process proceeds as follows.

1) For the molecule contained in the newly expanded node, query the RV database for all applicable RVs.
2) For each applicable RV determine all applicable reagents
3) Apply all unique combinations of RVs and reagents to the state contained in the newly expanded node.
4) For each new product generated, store in a list.
5) Increment the simulation depth reached by 1

6) Repeat steps 1-5 using the new list of products identified in (4) as starting points until a predefined depth is reached.
7) For all molecules which are at the predefined depth, score them using the scoring function. If molecules cannot be scored, they are given a score of -1.
8) Transform the list of scores generated into a single number using a mathematical transformation.
9) Pass the single number generated in (8) to the backpropagation routine.
10) Discard all molecules generated during the subroutine.

This subroutine is carried out sequentially for each node generated in the expansion stage. Step six requires the user to set how deep the subtree generated during the simulation should grow. The depth of the simulation is problem specific, and the default value set here is one. Step ten is implemented to reduce memory footprint. An alternative approach would be to transfer the molecules generated in the simulation to the main tree. One difference between the RVMCTS and the base implementation is that the score takes a continuous value between zero and one (as the scoring function is the Tanimoto coefficient), rather than a binary zero or one response. Another key difference between a standard MCTS and the RVMCTS is that the simulation step is a depth one full enumeration.

An additional deviation from the base implementation is the reliance on an enumerated simulation routine. Although there are two options that could be considered to develop a more stochastic simulation routine (randomly choose reaction vectors, and randomly choose generated molecules following the enumeration) the exhaustive approach was chosen in the preliminary study due to practical considerations. The reaction vector engine has been optimised to allow bulk storage and retrieval of reaction vectors. This means that the SQL queries and database have focused on retrieving all applicable reaction vectors and then enumeration. Thus, from an implementation perspective and to demonstrate the utility of the MCTS methodology, the enumeration approach was utilised. Simulation is explored in more detail in Chapter 8.

Finally, there is a limitation of the utility of the scores backpropagated when the lookahead depth is short for example in single step enumerative simulation. However, this limitation is mitigated by the shortness of the paths generated due to fragments being added rather than single atoms.

6.4.6 Backpropagation – Updating from leaf to node.

Backpropagation is the process in the MCTS where the path traced during the selection stage is updated with the information generated during the simulation phase. The backpropagation is the

final stage of a single iteration. The backpropagation routine used during the RVMCTS proceeds as outlined above for the general description of the algorithm.

As there are several nodes generated during the expansion routine, backpropagation updates the paths sequentially in the same order as the nodes were simulated.

At the end of backpropagation one iteration has been completed and the process either starts again or is terminated.

6.4.7 Post-processing

Following the completion of the RVMCTS, the final tree is exported via a complete recursive trace through the tree. The exporting process proceeds as follows:

1) Generate a list of all nodes generated during the search and the root node
2) For each node in the list write out the following: the state; the state of the parent node; the score of the state from the evaluation function; the UCT value of the node; the iteration in which the node was generated; and the timestamp of the node generated.

6.4.8 Prerequisites for the RVMCTS

The RVMCTS implementation requires a scoring function, a SM, a RV database, and a reagent pool. The scoring function and SMs are easily created and are not discussed further as they are problem-specific. The RV database and the reagent pool used in the initial implementation are described below.

6.4.8.1 Reaction vector database creation

The RVMCTS is designed for RBDD and applications in drug discovery. The following section discusses how a set of medicinal chemistry reactions were extracted from the literature and encoded in a RV database which is used as a pool of actions for the RVMCTS.

The top 15,000 reactions cited in the Journal of Medicinal Chemistry up to 2017 as determined by Reaxys (*Elsevier*, 2021) were exported. Following this, a validated cleaning process was carried out. Figure 6-11 outlines the general cleaning process used to generate the RV database. This process is implemented in KNIME and has been validated by Ghiandoni (Ghiandoni et al., 2019).

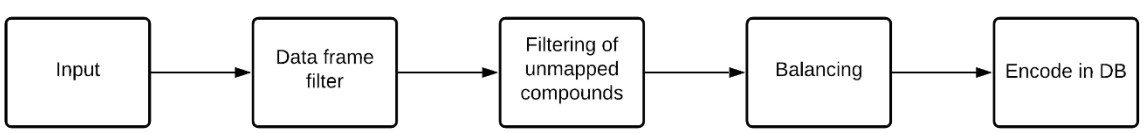


Figure 6-11 Database cleaning and encoding scheme

In Input, the reactions are read in and standardised as follows:

1) The reactions are split into SMs and products.
2) Both SMs and product molecules on both sides of the reaction are standardised.
3) The standardised SMs and products are combined into cleaned reaction SMILES.
4) The indigo reaction atom mapper is applied to the standardised reaction SMILES.

The data frame filter removes any missing data generated in the input standardisation process. Such as unreadable reaction SMILES.

In filtering, unmapped compounds, molecules which do not take part in the reaction-based on the atom mapping process are removed. The filtering process removes molecules such as solvents and catalysts. If the atom mapping process fails, then the reaction is removed.

In balancing, the reactions are balanced by passing them through a reaction balancing tool (Wallace, 2016), which removes salts and balances the number of carbon atoms. Balancing generates clean reactions ready for the encoding process.

In encode in DB, the clean reactions are taken, and RVs are calculated and stored in an H2 database with duplicates removed. The database encoding process contains further checks to ensure the RV is expected to work. The database encoding process has been extensively explored by Wallace (Wallace, 2016) and Ghiandoni (Ghiandoni, 2019). The reagents used alongside the RVs in the original clean reaction are also stored and are called internal reagents.

The 15,000 reactions mined from the Journal of Medicinal Chemistry were encoded into 4491 unique RVs. The database is named “JMedChem4.5k” and acts as the default pool of RVs and reagents for the RVMCTS.

6.5 Conclusions

In this chapter, a description of the basic domain-independent MCTS was given, following which a brief overview of the MCTS strengths and weaknesses as a method of the search was provided. As the base implementation of the MCTS is domain-independent, a brief overview of the current uses of the MCTS for chemistry was given, including usages such as retrosynthetic planning and atom-based de novo design. The latter half of this chapter turned to the detailed description of the

implementation of the RV-based Monte Carlo tree search (RVMCTS) for de novo design. To conclude, a description of the cleaning and encoding process of the RV database is provided. Chapter 7 focusses on the testing of the RVMCTS to understand the performance of the approach and how each subprocess influences the overall search.

Chapter 7 RVMCTS Ablation Studies

7.1 Introduction

The RVMCTS is the first implementation of the MCTS for RBDD, and this chapter focuses on a series of ablation type experiments to understand the utility of each sub-process within the RVMCTS. An ablation study is the removal of a component of a complex entity to observe its effect on the functionality of the entity. For example, in studies on understanding the function of different parts of the brain in early neuroscience, segments would be removed from lab animals to observe the change in their behaviour(Carlson & Birkett, 2017). The same principle has been applied here to understand the processes of the RVMCTS, by removal or modification of different parts of the RVMCTS, to establish a direct cause and effect relationship for different components of the algorithm.

Previous experiments using the MCTS algorithm have focused on empirical proof of the success of the algorithm or on theoretical contributions focussed on establishing operational bounds of the UCT equation (Browne et al., 2012). In the context of the chemistry domain, previous experiments using the MCTS in synthetic planning and de novo design have concentrated entirely on empirical performance compared to other established methods(Segler et al., 2018). Observed performance relative to a benchmark can provide an understanding of an algorithm's robustness and general practical use; however, purely empirical observation does not allow for an understanding of the parts of the algorithm which work best or areas that can be improved. To direct the development of the RVMCTS, a validated rediscovery benchmark has been created to systematically investigate the operational performance of the RVMCTS and then to investigate each component of the RVMCTS. The goal is to understand the contribution of each part of the RVMCTS algorithm and to establish a cause-and-effect relationship between each sub-process and the overall performance of the algorithm.

7.2 Methods

The rediscovery problem which forms the basis of the ablation studies refers to the rediscovery of a known final compound from a known SM. A working definition for the rediscovery problem type is provided below.

The recreation of a predefined goal molecule from a predefined SM within an allocated computational budget.

This definition is referred to as a working definition since there is no one agreed-upon definition of the requirements of a rediscovery problem, and more broadly, success for a de novo design approach is difficult to define (Brown et al., 2019).

7.2.1 Generating the benchmark datasets

A small set of validated rediscovery problems has been created as a benchmark. When constructing the rediscovery benchmark, a set of predefined "goal objectives" need to be created. However, this process could lead to bias being introduced. Benchmarking bias in this context is similar to selection bias in that a naive choice of problems can lead to a lack of differentiation in performance during the experiments. For example, if a synthetic scheme is of length one, then as the RVMCTS performs a full enumeration at the first level of the tree, if the required reagents and RVs are present in the dataset, it would solve all length one problems regardless of RVMCTS setup and configuration. Hence, a benchmark consisting only of single-step problems would not allow the performance of the algorithm to be investigated. In contrast, if the benchmark is composed of synthetic schemes of length 15+, then the opposite effect would be observed where the problems are far too difficult.

The process for creating the validated benchmark is shown in the block diagram below Figure 7-1:

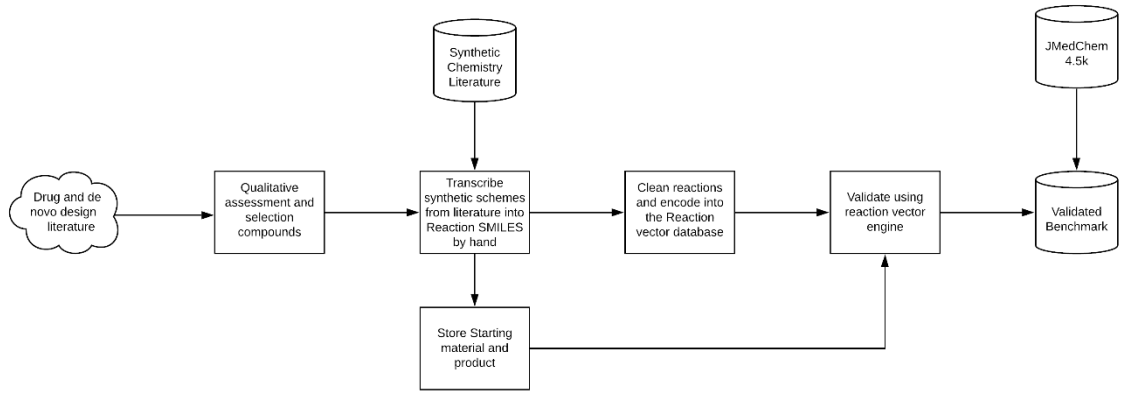


Figure 7-1 Block diagram of the creation of the validated benchmark. The process begins with selecting candidate molecules and their synthetic routes from the drug and de novo design literature by a qualitative assessment process. The reactions are cleaned and encoded using the process described in Chapter 6. The resulting RVs are then validated to ensure that the known product can be generated from the SM. The validated RVs are then combined with the JMedChem4.5k set to create the final validated benchmark.

Figure 7-1 describes the process of creating the validated benchmark. Each block represents a subprocess and the rest of this section will describe each subprocess in turn.

7.2.1.1 Starting molecule pool – Drug and de novo design literature.

The starting point for creating the benchmark for RBDD is deciding on a set of target molecules that will be used as the goal states. The dataset the molecules are taken from should be relevant to drug discovery, and the molecules must be synthesisable in the lab. The dataset chosen was the top 200 small molecule drugs by sales in 2018, which is a part of long-running series of compilations of the top drugs by sales starting from 2010 (McGrath et al., 2010). The dataset was chosen due to the diversity of compounds it contains, its direct medicinal chemistry relevance, and due to the popularity of the drug molecules; they will typically have a large, diverse collection of preparation routes from experimental to large scale process chemistry. Finally, the dataset has been used previously to test the robustness of RENATE (Ghiandoni, 2019), an RBDD system. When testing RENATE, the dataset was curated using an automated process to select molecules with a high chance to be designed using RBDD. This curation process returned 70+ molecules.

7.2.1.2 Qualitative assessment and subjective filtering

A subset of the top 200 molecules was chosen to act as a benchmark. The main criterion for selecting these was that they are theoretically solvable using the RV approach. The RV approach has previously been shown to have limitations with some types of molecules, especially those with complex scaffolds including complex heterocycle ring systems. For example, morphine (shown left in Figure 7-2) is a highly complex natural product with several fused ring systems and is not amenable

to synthesis via the RV approach, whereas, aripiprazole (shown right in Figure 7-2) is much simpler, with a heterocycle connected via an aliphatic ester to piperazine and to dichlorobenzene and it is expected that this would be synthesisable using the RV approach. It was also decided to limit the benchmark to molecules contain only specific atoms (C, N, O, F, Cl, Br, I, S, and P which are the most common atom types in drug-like molecules). Following visual inspection of the top 200 compounds based on these criteria, the molecules which were chosen are shown in Figure 7-3 below.

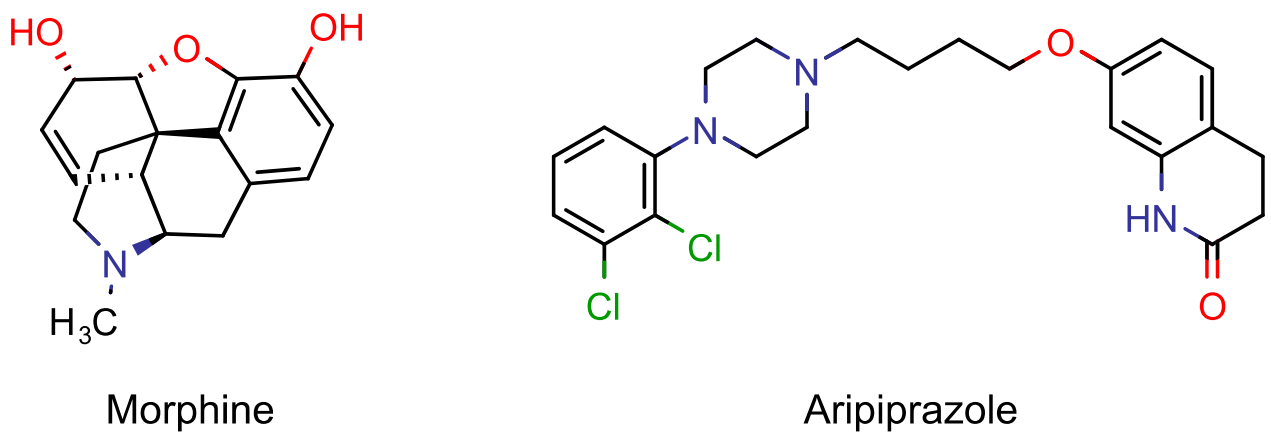


Figure 7-2 Morphine and aripiprazole. Morphine on the left would not be expected to be synthesisable using RVs. Aripiprazole, on the right, would be expected to be synthesisable using RVs.

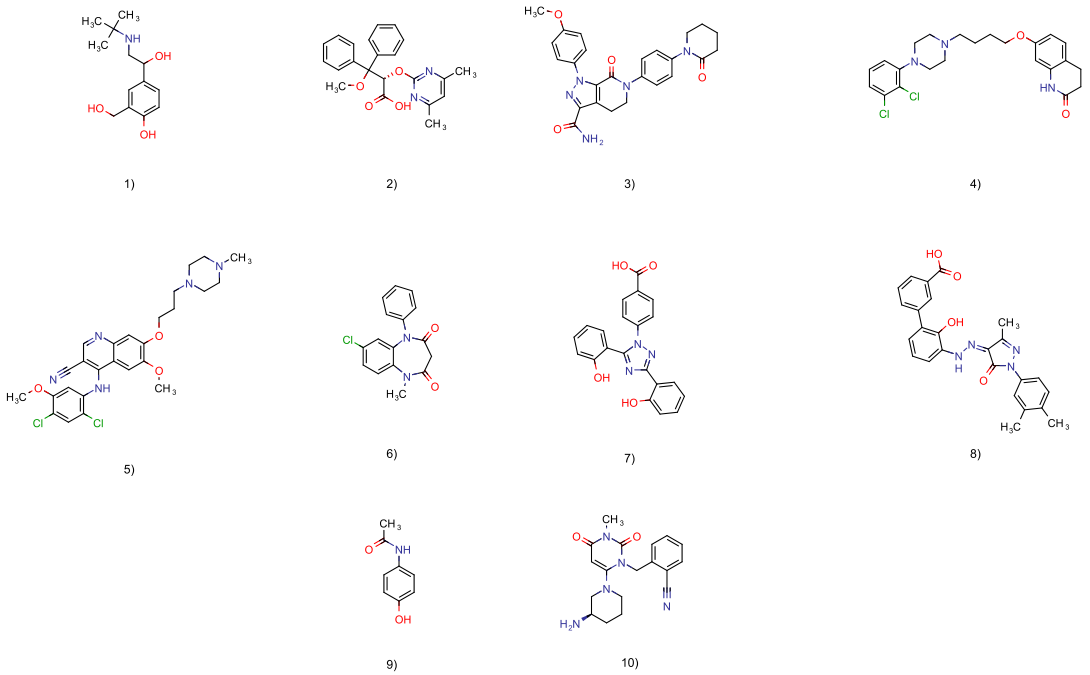


Figure 7-3. Ten candidate molecules with medicinal chemistry relevance taken from the top 200 small molecules by sales 2018 following a visual inspection. 1) Albuterol, 2) ambrisentan, 3) apixaban, 4) aripiprazole, 5) bosutinib, 6) clobazam, 7) deferasirox, 8) eltrombopag, 9) paracetamol, 10) alogliptin.

7.2.1.3 Transcription and encoding of reaction vectors

The ten candidate compounds were forwarded to the next step of the benchmark creation process, which is the selection of the synthetic schemes. The synthetic schemes chosen range between 2-6 steps. The schemes were chosen from the medicinal chemistry literature, the patent literature with yields, and the process chemistry literature.

The selection of routes from these literature sources is an attempt to guarantee that the schemes are possible in-silico. Each scheme was chosen based on amenability to RVs and a general focus on schemes without large structural changes such as deprotection, cyclisation, ring-opening, hydride shifts, rearrangements. Schemes are preferred if they are linear and not convergent.

The names of each compound, the length of the synthetic schemes, and their references are shown in Table 7-1.

Name	***Scheme length***	***Literature reference***
Albuterol	4 steps	(Wang et al., 2018)
Ambrisentan	4 steps	(Riechers et al., 1996)
Apixaban	6 steps	(Pinto et al., 2007)
Aripiprazole	3 steps	(Oshiro et al., 1989)
Bosutinib	4 steps	(Boschelli et al., 2004)
Clobazam	3 steps	(Kumar et al., 2016)
Deferasirox	2 steps	(Roatsch et al., 2019)
Eltrombopag	6 steps	(Revill et al., 2006)
Paracetamol	3 steps	(Ellis, 2002)
Alogliptin	3 steps	(Feng et al., 2005)

Table 7-1 Scheme lengths and literature references for the identified molecules which will form the benchmark.

The synthetic schemes were then manually transcribed into reaction SMILES. The reason for manual transcription is that frequently when writing synthetic schemes, chemists will include several reactions within one reaction arrow. So even though a synthetic scheme might be written as one step, in practice, the transformation may involve several sub reactions. These sub reactions might form intermediates that will not be isolated and therefore will be ignored by the synthetic chemist. Instead, a synthetic chemist will prefer to write out only those molecules which will be characterised. In contrast, when generating structures with RVs, all molecules and reaction steps are written out in full even if, in reality, they may only appear briefly during a synthetic scheme.

Following transcription, the synthetic schemes were then encoded into the RV database. The encoding process proceeds as outlined below in Figure 7-4, and as was discussed in Chapter 6.

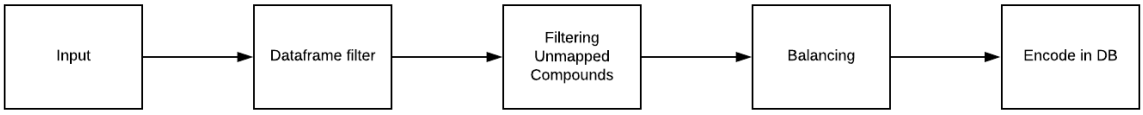


Figure 7-4 Process of encoding the database. The encoding process is provided in more detail in Chapter 6. For the creation of the database for the validation process the process is unchanged.

Following encoding, the next step was to validate the encoded RVs.

7.2.1.4 Validation of reaction vectors using a starting material

Validation is a straightforward process, as shown in Figure 7-5. The aim is to determine if the reaction can be reproduced algorithmically.

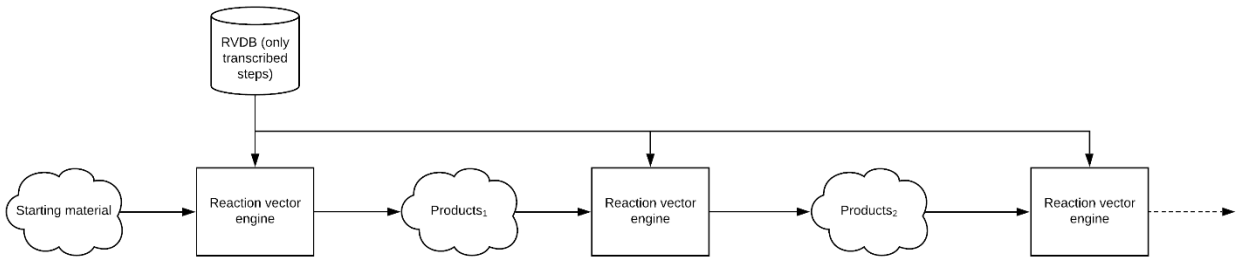


Figure 7-5 Sequential validation process. The SM is passed to the RV engine, and all applicable RVs are applied. The expansion process generates the pool of products which is then passed into the RV engine. Again, all applicable RVs are applied generating products$_2$. This process repeats until the end of the synthetic scheme. Thus, there would be two expansions for a synthetic scheme of length two.

The SM was that used in the literature scheme, the reagents were those which exist within the internal reagents of the encoded RVs (Chapter 6). The expansion step was repeated for the same number of steps as the length of the expanded, cleaned synthetic scheme. Therefore, if the scheme is five steps long, five sequential full enumerations were carried out. The final step was to determine if the known product was obtained.

The results of the transcription, encoding and validation of the reaction schemes are shown in the following Table 7-2, where Y means successful validation and N means unsuccessful. A depiction of the successfully transcribed schemes are given in appendix B. The note column provides a potential reason for why the scheme failed to encode or failed to validate. However, it was not always possible to determine the precise reason. First, encoding is a complex process described in Chapters 4 and 6 and in more detail by Wallace (Wallace, 2016). Briefly summarised here, the reason for failure in the encoding process is that following fragmentation of the molecule into constituent parts, the fragments are then reconstructed using a recombination path. Unfortunately, when this recombination occurs, there is a time limit set on how long the recombination of the fragments to the molecule will take (<30s). The exact reason for failure to encode is not provided and instead is

simply attributed as a timeout error. Second, failure can occur on the application of a RV for a similar reason as outlined above. When the SM is fragmented, the product is built using the applicable RVs recombination path; however, this recombination may not generate a valid molecule at the end of the path.

Name	*Successfully validated*	*Note*
Albuterol	N	Fails to encode, step two Bromine SN2
Ambrisentan	Y	Pass
Apixaban	N	Fails application due to step three alkene halogenation coupling
Aripiprazole	Y	Pass
Bosutinib	N	Fails application due to the first stage ring-closing reaction. The ring-closing reaction contains a nitrogen heterocycle
Clobazam	N	Fails to encode step two due to Chlorine SN2
Deferasirox	N	Fails to encode due to recombination. Nitrogen cyclisation
Eltrombopag	N	Fails to encode step one due to the nitro reduction
Paracetamol	N	Fails to encode due to the nitration step
Alogliptin	Y	Pass

Table 7-2 The selected molecules, validation outcome and attribution of possible failure.

The reactions for the validated schemes were then encoded in a larger database alongside the JMedChem4.5k RV set, which acts as noise. By introducing noise to the RV database, the problems were made more realistic by increasing the number of applicable RVs for each molecule while guaranteeing that the correct synthetic route was present in the database.

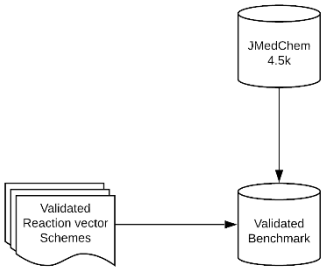


Figure 7-6 Final step creating the Validated benchmark

Finally, a potential concern arising from having only three schemes in the benchmark relates to the generalisability of the results. However, it was felt that three examples are sufficient to evaluate the method on this specific type of de novo design problem posed in this chapter which is the ability to regenerate a known target molecule based on similarity to that target. Furthermore, this scale of benchmark for this type of problem is typical of the way in which other novel de novo design approaches have been evaluated in the literature(Brown et al., 2019; Segler et al., 2017; Zaliani et al., 2009). Different types of design problems and scoring functions are explored later in the thesis. These results however, do not allow any reasonable prejudgement of the quality of future generated solutions for different scoring functions.

Reaction map

To visually demonstrate that the reaction transformations that have been hand encoded and transcribed into the validated database are relevant medicinal chemistry transformations, a map of reaction space was built. The validated database as shown in Figure 7-6 contains reactions from the Journal of Medicinal Chemistry and the hand encoded RVs, therefore, the visualisation shows the relative proximity of the hand encoded reactions to the literature medicinal chemistry reactions. The process of building the reaction map used the following routine shown below:

1) Convert cleaned reactions into RV fingerprint (Ghiandoni et al., 2019)
2) For each reaction, classify it using the SHREC ontology (Ghiandoni et al., 2019)
3) Perform a PCA to reduce the number of dimensions of the reaction descriptors down to 256 dimensions
4) Perform a UMAP supervised dimensionality reduction (McInnes et al., 2018) to two dimensions using the reaction labels from the SHREC ontology with the 256 dimension PCA components.
5) Plot the resulting 2D dimensions in a scatterplot

A visual representation of the process is shown below.

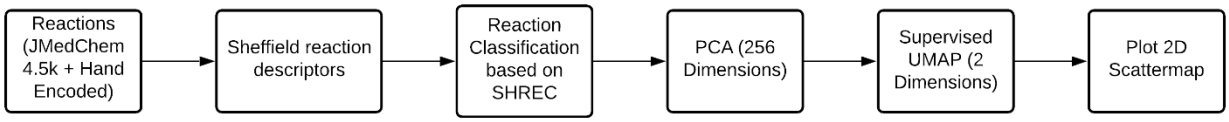


Figure 7-7 A visual representation of the workflow to generate the reaction map

The reaction map is shown in Figure 7-8 with different reaction classes shown in different colours. The reaction map shows that the reaction classes have separated well based on the SHREC labelling.

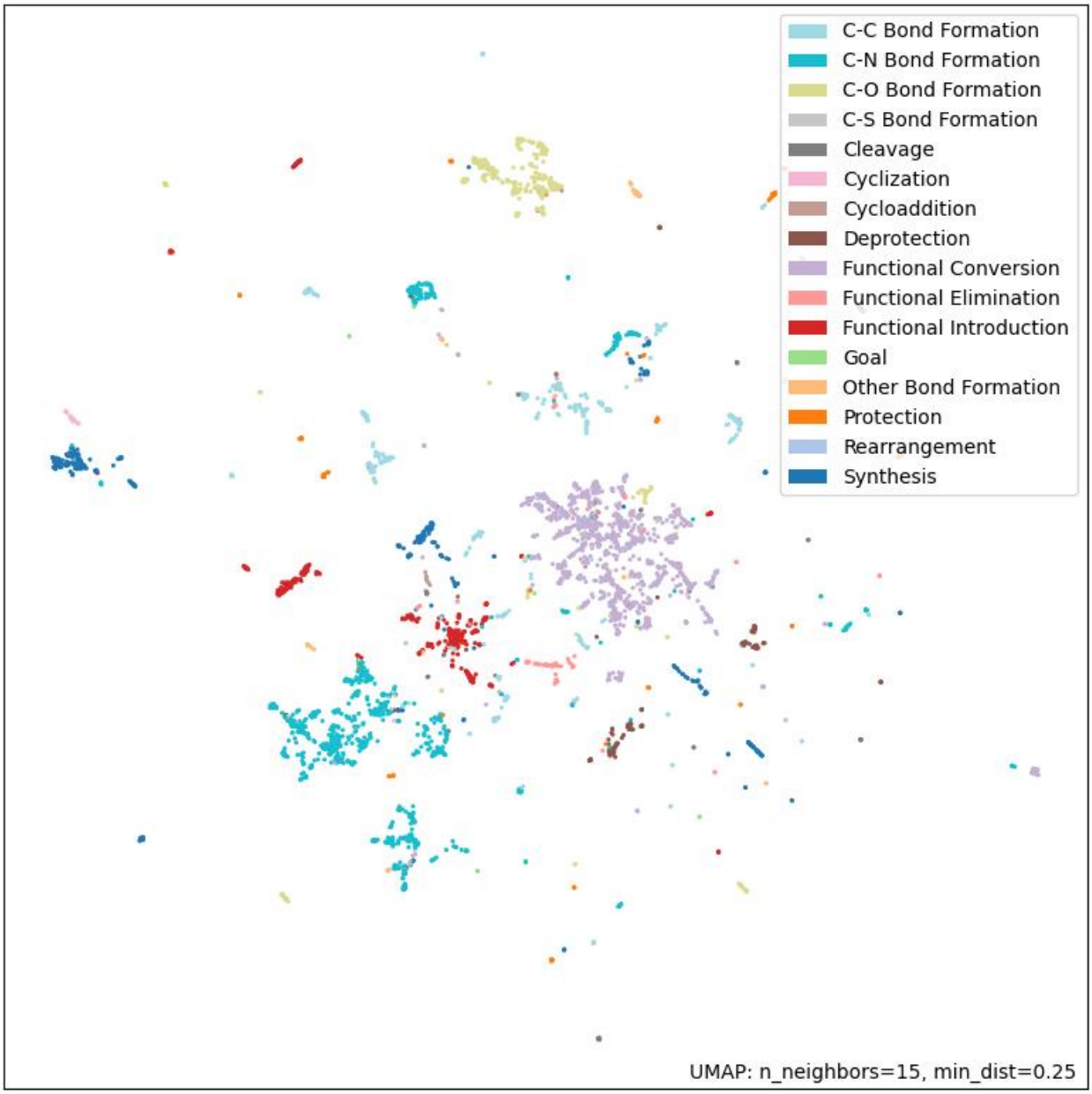


Figure 7-8 Map of reaction transformation space. This map was generated using a PCA-Supervised UMAP process. Clear separation can be seen between clusters of different reaction types.

Figure 7-8 shows that the reaction transformation space that is encoded is diverse; however, the colour map chosen makes it hard to identify where in the space the goal transformations exist. Figure 7-9 below is the same colour map; however, the goal transformations are coloured in brown, and the reaction transformations present in the JMedChem 4.5k set are coloured in blue. The reaction map shows that most of the goal reactions are embedded in the clusters of the reaction transformation space.

Figure 7-9 The same reaction map generated as Figure 7-8, with a contrasting colour map to highlight where the goal transformations are in reaction transformation space.

In total, there are ten RVs that are encoded from the seeded reaction schemes; however, in the above plot, there are only eight visible dots. This is due to two reactions having the same descriptors and they are therefore positioned on top of each other.

Scoring function

The scoring function used in the RVMCTS is Tanimoto similarity (Tc) to the goal molecule using the Morgan Fingerprint radius 2, hashed and folded to 2048 bits as implemented in RDKit (Release 2020.03.4).

The choice of the Tanimoto coefficient is as a replacement for a more complex scoring function that could exist; it is not uncommon to use the Tanimoto coefficient as a proxy for a QSAR model (Brown et al., 2019)

Table 7-3 below shows the SMs and their corresponding similarities to their respective goal molecules.

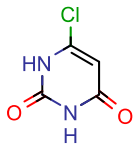
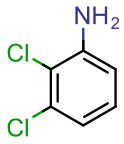
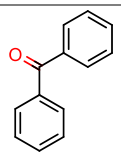
Problem	SM	Tanimoto similarity
Alogliptin		0.08
Aripiprazole		0.18
Ambrisentan		0.21

Table 7-3 problem name, SM depiction, and tanimoto similarity of the SM to the final goal structure

7.2.1.5 RVMCTS Parameters

The RVMCTS requires several parameters to be set Table 7-4 below is a summary of those parameters.

Parameter	Description
Total Iterations	Number of iterations the MCTS will undergo before completion
Exploration Constant	The parameter which scales the amount of exploration that is occurring
Simulation type	Dictates what type of simulation occurs
Backpropagation type	What information is passed up the tree from the leaf node.

Table 7-4 Parameters used in the RVMCTS, total iterations, exploration constant, simulation type, backpropagation type.

Deciding on appropriate parameter values for the RVMCTS in advance is difficult as good parameter ranges in the MCTS will vary substantially between domains and specific problem instances. The two primary considerations when deciding the parameter values for the RVMCTS in this experimental setting were the time that a single search takes to complete and the probability of success. First, time is important because long searches can last weeks if a poor choice of the number of iterations is made. Second, the parameter selection has to lead to a reasonable probability of success; for example, if a poor choice is made for the exploration constant, it could be that the final goal can never be found in the allocated computational budget. Hence conservative estimates are required for the base parameters, and these were determined through a mix of trial and error and domain knowledge. Finally, parameter optimisation is a large part of tuning the MCTS for optimal performance. While optimal performance is the desired goal within the context of solving real-world problems, the purpose of these ablation studies was to observe how the RVMCTS performs when individual parts are removed.

Visualisation of the search space and the trees

The processes of visualising the molecules generated and the tree growth during the search are important qualitative tools to understand how the ablation of different components leads to different search performance. The visualisation of the tree is focused on the shape of the tree, whereas visualisation of the chemical space of molecules generated aims to show how the molecules are distributed over the space and whether there are clusters of molecules formed.

Visualisation of the generated chemical space

To visualise the chemical space that has been generated, a similar process to that used to visualise the reaction transformation space in Figure 7-8 was employed. The process of visualising the chemical space that is generated is shown in Figure 7-10 below. First, the molecules were converted to their Morgan fingerprint representations. These representations were then reduced to 256 dimensions using a PCA. The reduced representations were then passed into the UMAP method, where the number of dimensions was then further reduced into two components. These two components were then plotted on a 2D scatter.

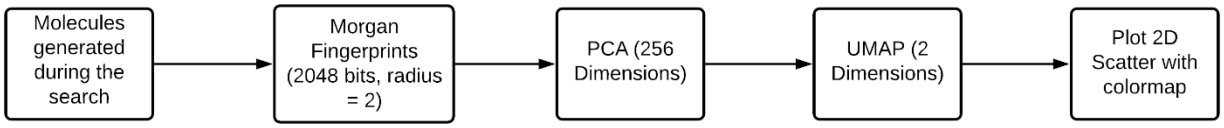


Figure 7-10 The process of building a 2D representation of the chemical space that is generated during the search.

Tree visualisation

In contrast to chemical space visualisation, direct visualisation of the tree generated during the search is more challenging. This is because the tree will contain many thousands of molecules each of which may participate in one or more reactions. This leads to the number of edges exceeding the number of nodes. Therefore, to cope with the large amount of data to visualise, the hierarchical layout used has variable edge thickness. The more times a node has been visited, the thicker the edge between the node and its parent. This reduces the visual clutter of the diagrams. Furthermore, the nodes are sorted based on their score at a given level of the tree. The workflow for the generation of the tree visualisations is shown below.

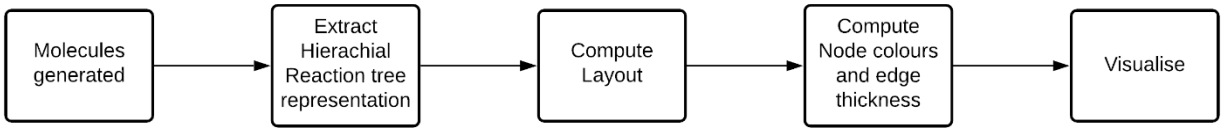


Figure 7-11 The tree visualisation process, first the hierarchical reaction tree representation is extracted, the layout is then computed using the hierarchical layout within EoN, The node colours are computed along with the edge thickness based on the similarity and the node visit count, respectively. Finally, the tree is visualised and exported for viewing.

Experimental workflow

The workflow for the ablation studies was first to establish a baseline performance for each target molecule based on the RVMCTS with all components and then to systematically remove each component to observe the change of the RVMCTS compared to the baseline or control.

A visual description of the experimental setup is outlined below.

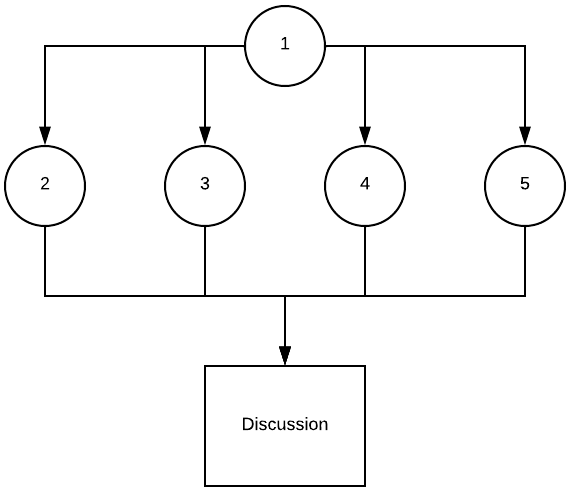


Figure 7-12 Flow chart showing how the experimental process of the ablation studies was carried out. Firstly, a series of controls was created in experiment one, and then in each proceeding experiment, a single part of the RVMCTS was ablated. 2- Ablated selection, 3- Ablated expansion, 4- Ablated simulation, 5-Ablated backpropagation.

The ablation studies consisted of removal of selection only, the expansion only, simulation only, and finally backpropagation only. All the experiments were run for all three validated rediscovery problems in the RVMCTS benchmark.

Experiment one: - Construction of controls

The parameters used for the control experiments are given in Table 7-5 and are based on the default values described in Table 6-4.

Parameters	**Value or Setting**
Total Iterations	50
Exploration Constant	0.001
Simulation type	Depth one full enumeration
Backpropagation type	Continuous

Table 7-5 RVMCTS Parameters and their corresponding base values used during the search.

The reasoning for the above parameters is the following. The total iterations were set to 50, which is a low value compared to other domains, but this is due to the typically high computational cost of an iteration. A single iteration consists of selection, expansion, simulation, and backpropagation, with expansion and simulation being the stages that generate molecules. The expansion step involves a full enumeration of all applicable RVs on the molecule represented by the selected node (Ghiandoni et al., 2020) and the simulation step then involves a full enumeration of all child molecules. The exploration constant was set at 0.001 , as highlighted previously, the purpose of these parameter values was to provide a baseline. It was not expected that they would be "optimal".

Variance in the RVMCTS searches arises if a tiebreak occurs during the selection process, that is, if two or more nodes at the same level have the same value of the UCT. This is because a tie break is broken at random. If the RVMCTS has many tie breaks during the search, then it would be expected that the searches would have a wide amount of variation in their final scores and the path taken through the chemical space. In contrast, if the searches have few tie breaks occurring, then the opposite would occur. To understand the effect of variation during the search, ten searches were run with the baseline parameter values, and the performance over the search was averaged, and the variation noted.

The progress of the RVMCTS was observed by recording the maximum similarity to the goal with increasing number of iterations. It was predicted that due to the RVMCTS being a fragment-based approach, increases in maximum similarity would occur in a stepwise manner, where each step is the application of a single RV which progresses the search closer to the goal.

Experiment two: Ablation of Selection

Selection experiments

The selection step involves selecting the next node to expand thereby generating a subtree. The node that is selected is determined by the UCT equation. The ablation study consisted of omitting the UCT scoring and selecting the node to expand at random.

Experiment three: Ablation of Expansion

Expansion experiments

Expansion is the generation of new nodes within the tree. The process, as defined in Chapter 6, is a full enumeration. That is, all applicable RVs are applied to a given node. Full enumeration generates all possible products which are stored in separate nodes. Removal of the expansion step would result in a search that does not generate any nodes. Hence, for the ablation studies, different

amounts of expansion were investigated with 25%, 50%, and 75% of total products removed at random, respectively.

Experiment four: Ablation of simulation

Simulation is the process of ascertaining a node's value through a look-ahead search. New states generated in the simulation are scored, and a value is backpropagated up the tree. Simulation is the most computationally demanding part of the RVMCTS. It is therefore imperative that the usage of a simulation is justified, as it will ultimately add a large amount of computing time to the overall search. To ascertain the value of the simulation component of the RVMCTS, it was removed from the search, and scoring was based on the expanded node only.

Experiment five: Ablation of backpropagation

Backpropagation is the process by which information is passed back up the tree. Backpropagation reinforces paths in the tree from the root node down to the newly expanded node. The information that is reinforced in the tree will either increase or decrease the chance of nodes in the path being selected on the next iteration. The ablation study removed the backpropagation step so that there is no updating of the paths in the tree.

As the searches are expected to have a small amount of stochasticity present in the search due to ties, ten replicates of each search configuration were carried out.

7.2.2 Analysis

Analysis of the MCTS was broken down into four categories: algorithm performance, resources used, robustness, and qualitative assessment of the tree (Barr et al., 1995). Algorithm performance considered performance on the problem and included: the maximum score found; the average score; and properties of the score distribution, such as kurtosis and skewness. Resources used was measured by the number of molecules generated during the search and the time that the search took. Finally, robustness attempted to understand how generalisable the results from the search are; for example, are the results that are generated going to be consistent across repeat runs.

7.2.2.1 Algorithm performance

Algorithm performance was represented by eight measures:

1) Maximum similarity score found during the search
2) The average score of the whole tree found during the search
3) The skewness of the scores generated during the search
4) Kurtosis of the scores generated during the search
5) Number of iterations required to reach the maximum score
6) Depth of the tree reached during the search
7) Diversity of molecules generated during the search
8) Branching Factor

7.2.2.2 Resources used

Resources used was represented by two measures:

1) Total time taken for the search to complete
2) The number of molecules generated during the search.

7.2.2.3 Robustness

Robustness was represented by two measures:

1) The standard deviation of maximum scores across the ten runs.
2) Percentage of RVs used from the validated database

7.2.2.4 Qualitative assessment

1) Visualisation of Search Trees
2) Chemical space maps
3) Molecule inspection

7.3 Results and discussion

Results are presented below for the baseline and ablation experiments considering the different performance measures in turn.

7.3.1 Algorithm performance

7.3.1.1 Maximum score

The maximum scores found in the tree across the replicate runs are shown in Figure 7-13 for the three goal molecules. The baseline search performed well in the case of alogliptin, with the mean maximum similarity of 0.737, but worse in the cases of aripiprazole and ambrisentan, with mean maximum similarities of 0.468 and 0.333 respectively. The decrease in the performance is due to the differences in the search space that is being explored.

In the RVMCTS algorithm, the search space influences success due to two factors: 1) Accessible high scoring local minima close to the starting point, 2) Plateaus in the search space. Considering factor one, molecules that have high scores at the shallower depths of the tree will be favoured over molecules with high scores that are deeper in the tree. This is a well-known limitation of the basic MCTS algorithm (Browne et al., 2012). Plateaus in the search space influence the baseline performance due to the added computational time to determine which molecule on the plateau would lead to the best region of the search space. Plateaus lead to nodes having similar UCT scores, and the RVMCTS spending many iterations trying to determine which node is the best path. For example, there might be two nodes in the tree with one the result of a chlorination and the other the product of a bromination. The difference in the scores of these two nodes will be minimal. The RVMCTS has no understanding that these nodes come from the same reaction class, and a large amount of computational expense will be used to deduce whether the chlorination or the bromination is better. The success in the case of alogliptin is due to the high scoring intermediates and the lack of scoring plateaus in the search space.

Examination of the ablation of selection, expansion, and backpropagation experiments yielded surprising results. Ablation of selection removed the UCT equation so that selection was replaced by a random choice amongst the child nodes of the selected parent node. For the aripiprazole problem, removing the UCT equation gave improved performance relative to the baseline. The poorer performance seen when the UCT equation was used for selection suggests that the search is getting stuck in a local optimum. Trapping into a local optimum does not occur when the selection is ablated, and a higher maximum score is reached.

The ablation of expansion comprised three separate experiments, 75% retained, 50% retained, and 25% retained. The three separate configurations yielded similar performance which, although

showing a slight degradation as larger percentages were removed, was within the standard deviations of the replicates of the baseline. Thus, random removal of nodes during the search does not significantly affect the best score achieved.

Ablation of simulation outperformed the baseline substantially on the alogliptin and aripiprazole problems and performed the same as the baseline on the ambrisentan problem. On the alogliptin problem, the target molecule was found when the simulation was ablated. The improved performance suggests that the simulation component of the search has not aided the growth of the tree. The improvement can be ascribed to two reasons: considerable variation in scores of the nodes and a greedier search. When simulation is ablated the node scores are based on the molecule represented by the node, in contrast to when simulation is used, when the node scores are based on the average score obtained in child nodes during simulation. The latter is not likely to vary much, as discussed above. A large variation in scores of the nodes leads to greater variation in the UCT scores across each level which means that selection over successive iterations is directed strongly compared to the baseline. This means that the tree will grow deeper and is likely to lead to higher scoring molecules. In addition, the greater variation in the scores seen when simulation is ablated leads to a greedier search.

Ablation of backpropagation performed the same as the ablation of selection due to the search being random in both cases. As backpropagation updates nodes, removal of backpropagation prevented any node updates from occurring. Thus, the observed performance of the ablation of backpropagation search is equivalent to ablation of selection.

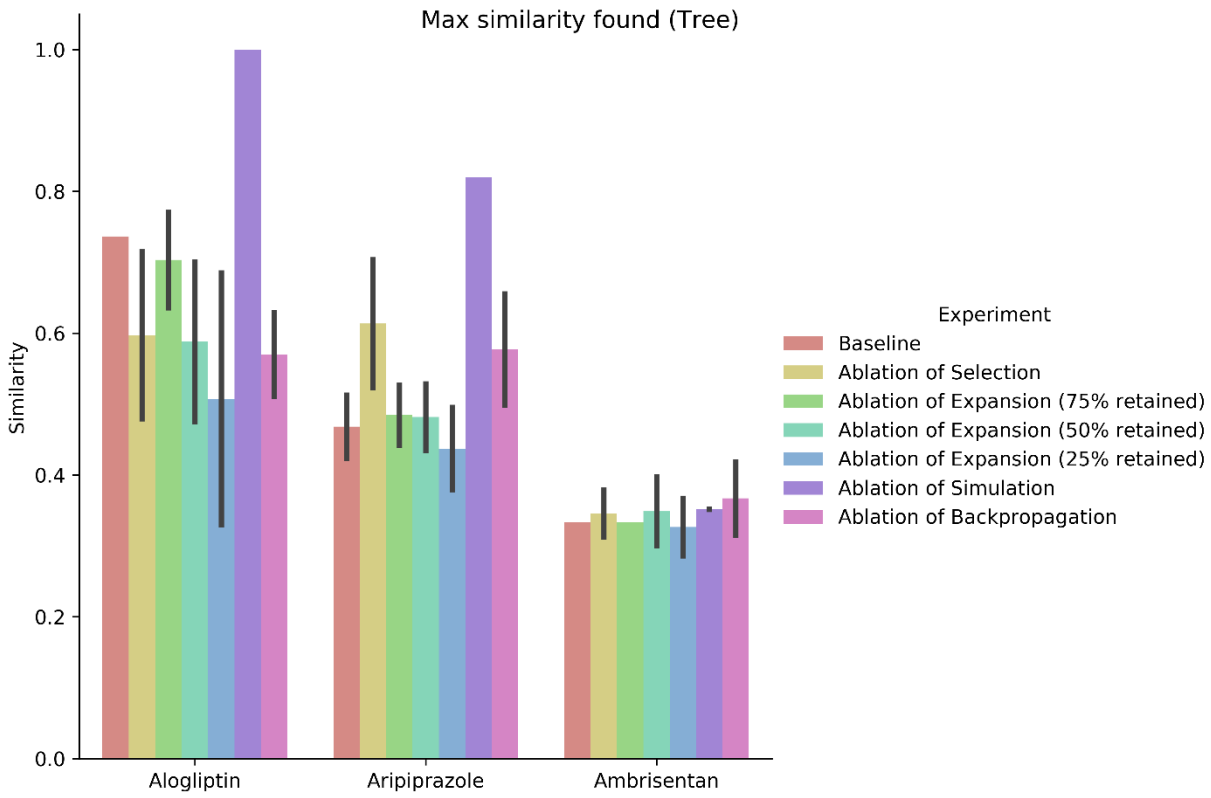


Figure 7-13 Maximum similarity found in the tree. The height of the bar is the mean; the black bars are ± 1 standard deviation of the data. Some bars have an SD of zero, and therefore no black bar is present.

In the RVMCTS, molecules are generated in both the tree and the simulation phase. In terms of algorithmic performance, it is helpful to explore the maximum similarity score generated during the simulations. It was expected that the similarity scores obtained from the simulation would be higher than those found only in the tree with the increase due to the increase in the depth of nodes. The maximum similarity scores found considering both the trees and the simulations for the different ablation experiments are shown in Figure 7-14.

Focusing first on the baseline, in all three cases, the maximum similarity score was higher for molecules generated during simulation than those in the tree. The baseline successfully solved the alogliptin problem and performed better across the aripiprazole and ambrisentan problems and the performance was equivalent to the ablated simulation search. However, although molecules with relatively high scores were observed during simulation, the simulation process does not direct the search into promising regions of the search space.

In ablation of selection, removing the UCT equation led to increased performance in the ambrisentan problem. The improved performance suggests that the search reached a local minimum during the ambrisentan problem and was trapped. As described above, when selection was random,

there was a reduction in trapping due to local minima, which, when coupled with the increased depth seen by the simulation step, gave rise to the improved performance. It is interesting to note that this suggests that the ambrisentan and the aripiprazole problems are deceptive. Deception is when a search algorithm is tricked by the scoring function. Further evidence for the deceptive nature of the ambrisentan problem is the reduced performance of the ablation of simulation tree compared to the random searches of ablation of selection.

When examining the molecules generated in both the tree and the simulation for the ablation of expansion experiments, greater variation was seen across the replicates. Greater variation is not surprising as random removal of nodes leads to removing intermediates which can lead to high scoring regions of the search space. Ablation of backpropagation performed similarly to the ablation of selection and demonstrated that random search led to competitive performance compared to the baseline.

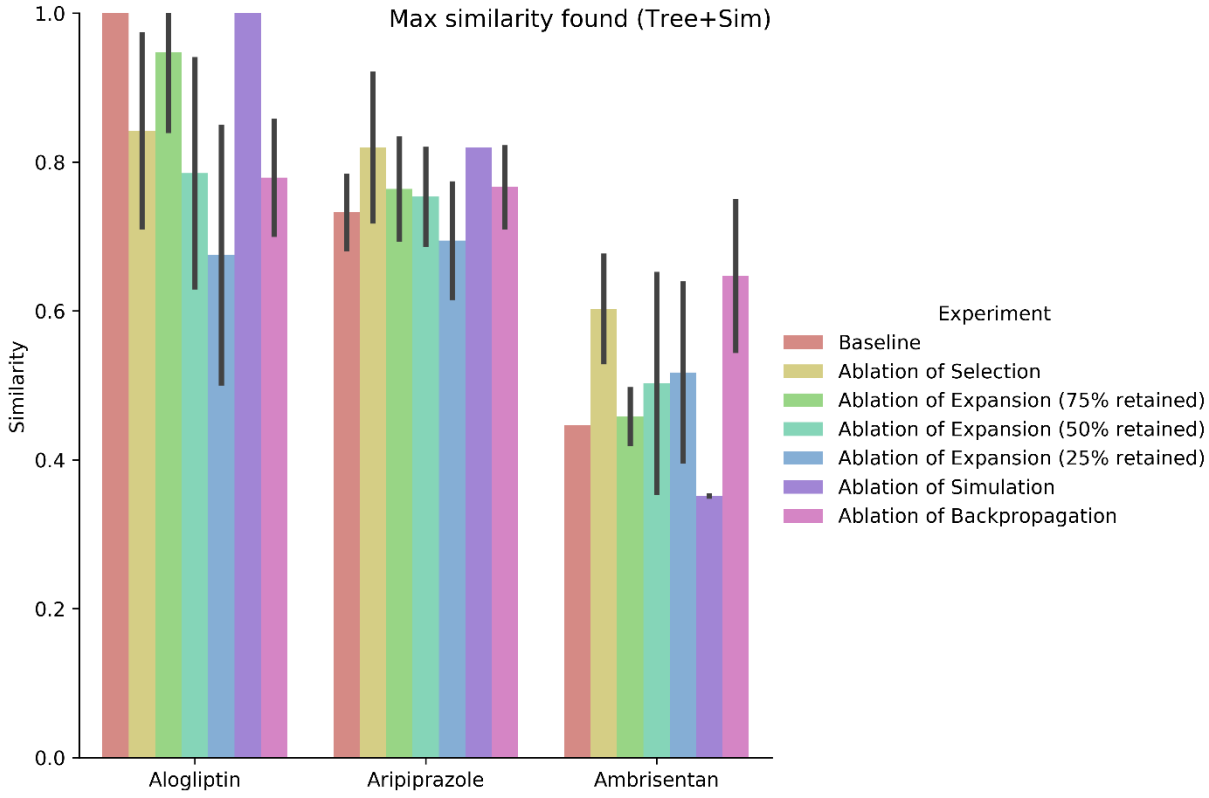


Figure 7-14 Maximum similarity found (Tree+Sim). The height of the bar is the mean, and the black bars are ± 1 standard deviation.

7.3.1.2 Average score

The average score is a measure of the average quality of the molecules generated during the search. Figure 7-15 shows the average score of all the molecules found within the trees. The average score helps understand how the distribution of scores is shifted for the different experiments, although it does not capture the shape of the distribution. It is important to note that the UCT equation is constructed to ensure that a degree of exploration occurs, and, therefore, it would not be expected that the entire distribution of the tree would shift to the goal molecule.

The baseline average scores were similar across all three problems. The results for the ablated selection, expansion and backpropagation experiments were all similar to the baselines. In contrast, ablation of simulation led to improved performance overall and outperformed the baseline on all problems. The increased performance by removing the simulation component led the search to be greedier and focused on deeper regions of the search space with higher scoring molecules. Ablation of simulation again highlights that although good molecules are generated during simulation, the simulation component does not direct the search towards high scoring regions of the search space.

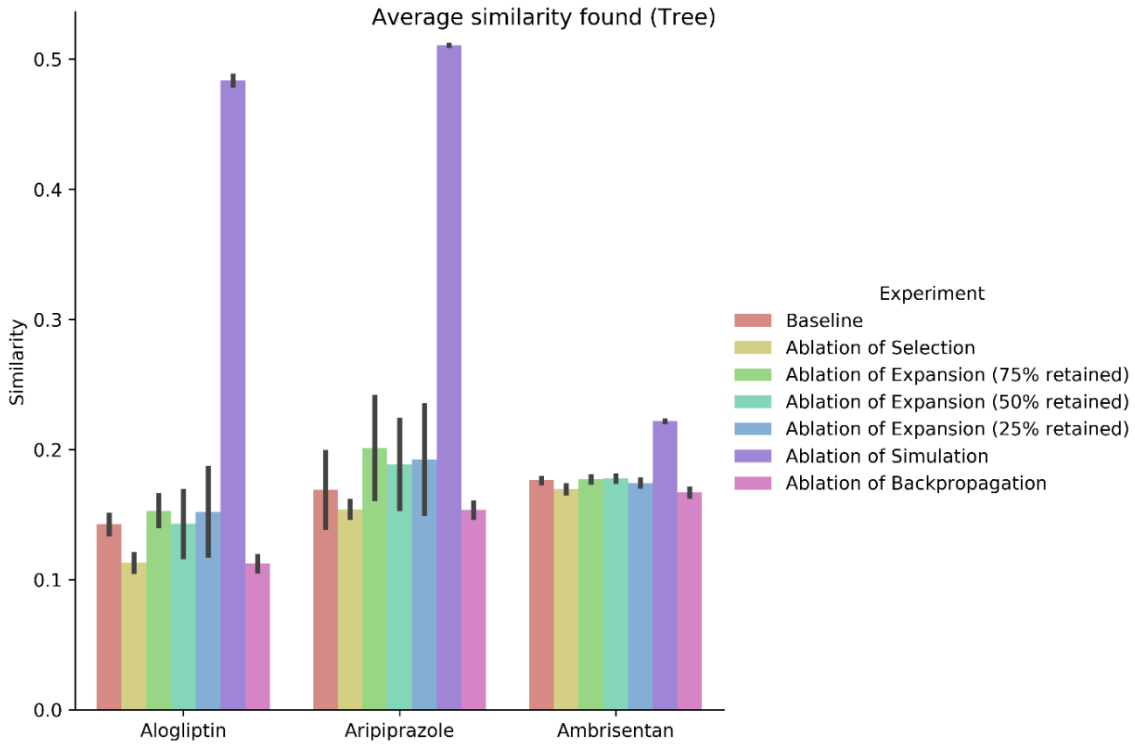


Figure 7-15 Average (mean) similarity of the tree. The height of the bar is the mean, the black bars is ± 1 standard deviation.

7.3.1.3 The skewness of the distribution

Skewness is a measure of the direction of the tail of the distribution. Here, positive skewness suggests the tail points towards a value of one (maximum similarity), whereas negative skewness implies the tail points to 0. If a distribution shifts from low values to high values, the skewness of the distribution will first start with positive skewness (a tail point towards a value of one) with few molecules close to the goal then, over time, the skewness will decrease until the distribution shifts towards having a tail with a negative skew, that is, a few molecules with low scores. A method that generates a distribution with a negative skew suggests that the tree has moved through the search space to a region of interest. A positive skew suggests the search has not moved through the search space towards the region of interest.

First, the baseline searches have decreased skew from highly positive to slightly negative across alogliptin aripiprazole and ambrisentan, respectively. As the skewness of the baseline decreased, the skewness shifts towards being more negative, suggesting that overall molecules are generated that are higher scoring than the SM. The skewness is consistent with the baseline for the alogliptin and aripiprazole problems for ablation of selection. However, skewness is slightly more negative for the ambrisentan problem which can be attributed to the rougher search space of ambrisentan. As discussed above, when selection is ablated the search is random, however the skewness is more negative for ablation of selection than the baseline for the ambrisentan problem. This would suggest that the ablated selection search has moved to a deeper depth. Hence, removal of the UCT equation in rough search spaces can lead to an overall increase performance. However, the standard deviation observed across the problems is much higher. The high standard deviation is due to the sensitivity of the skewness metric to sample size.A similar pattern is observed for ablation of expansion. Ablation of simulation outperformed the baseline and had a negative skewness throughout all three problems. Ablation of simulation shifted the distributions closer to the overall goals suggesting removal of simulation led to greedier searches and deeper trees and thus high scoring molecules. Ablation of backpropagation yielded the same performance as ablation of selection.

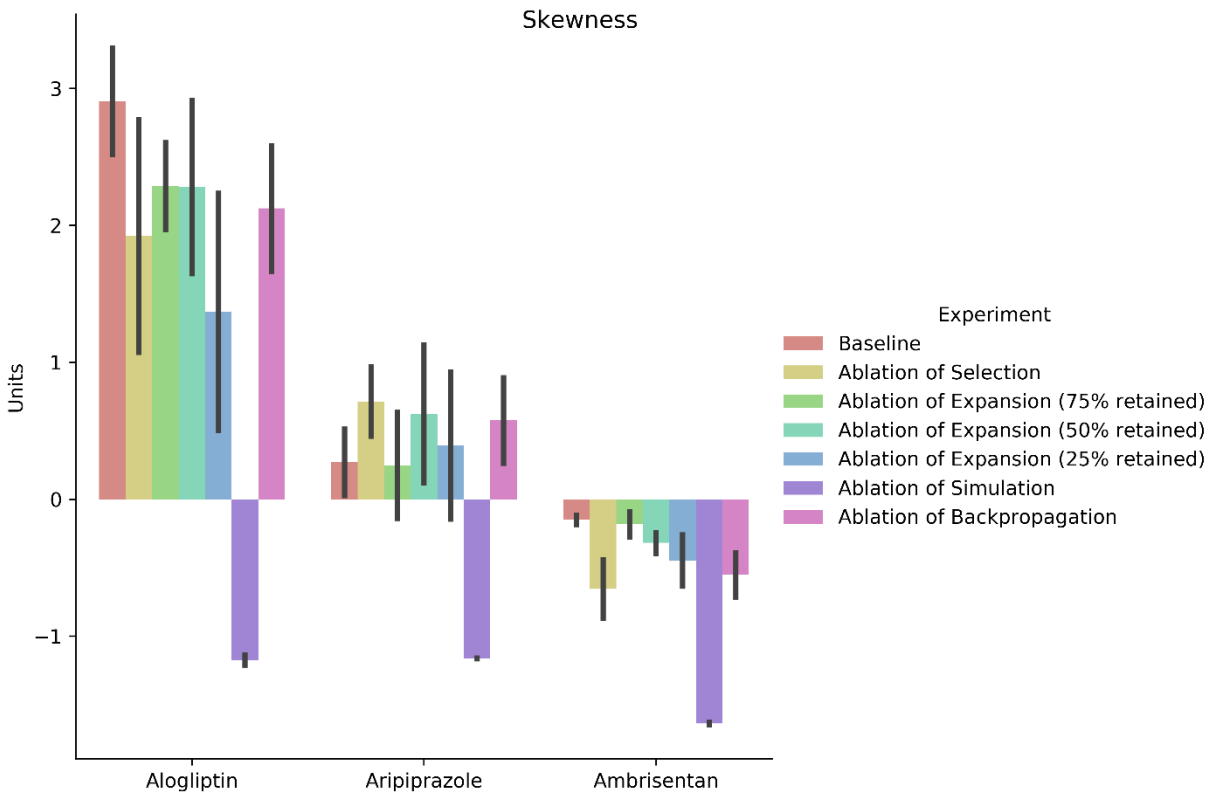


Figure 7-16 Skewness of the distribution of similarities generated during the search. The height of the bar is the mean; the black bars are ± 1 standard deviation.

7.3.1.4 Kurtosis of the distribution

Kurtosis is a relative measure of the data distribution compared to a normal distribution. In particular, kurtosis measures whether the data is present in the tails of the distribution. For example, a normal distribution of solutions would have a kurtosis of 0; if the data distribution is contained in the tails, then kurtosis is greater than 0, if the data distribution is contained in a prominent central peak with little data present in the tails, the kurtosis would be less than 0. The kurtosis of the distribution reveals whether the search has focussed strongly on one region of the search space or is more spread and diverse. The results in Figure 7-17 show a progressive shift in kurtosis across problems. The decrease in kurtosis suggests that the searches are exploring more in the alogliptin problem, which has data present in the tails of the distribution (high scoring compounds), and that the search is not focussing on one region of the space (regardless of average score). Interestingly, the standard deviation in kurtosis decreases as well across the problems which can be related to problem difficulty. However, observation of kurtosis does not reveal whether the solution quality is being affected directly.

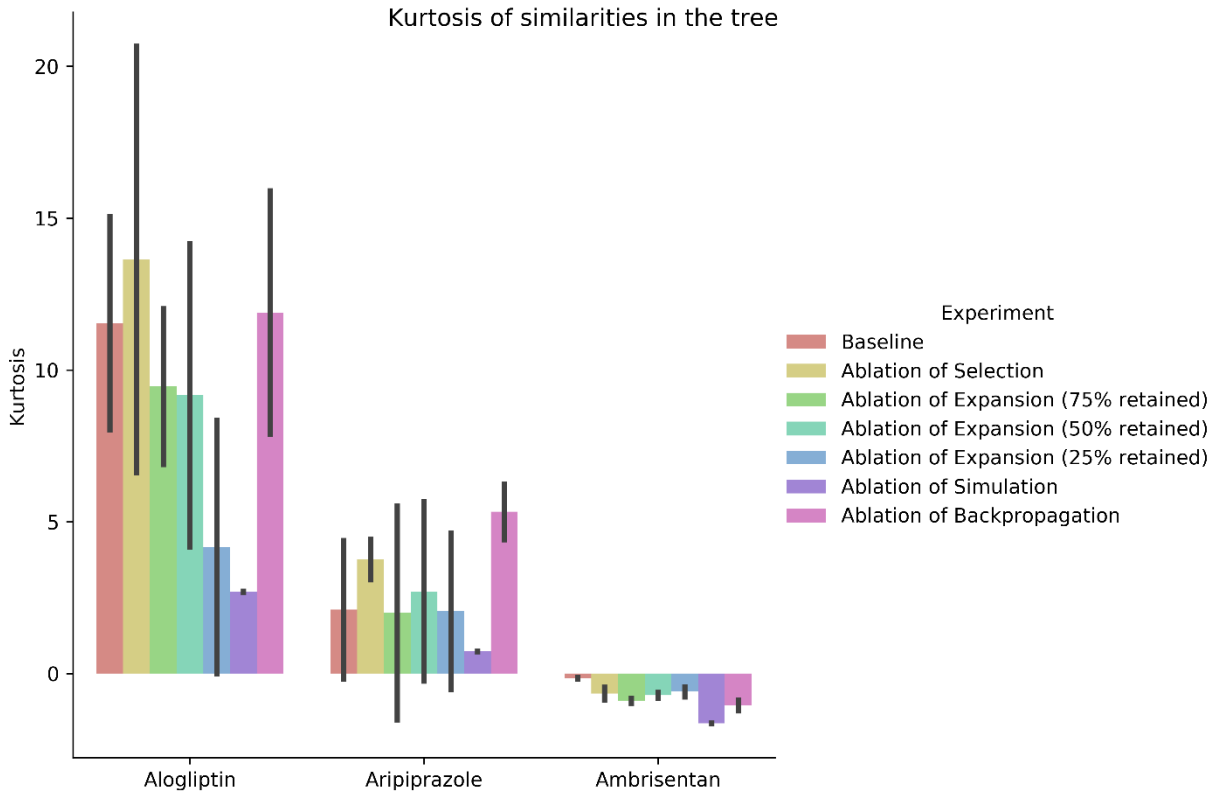


Figure 7-17 Kurtosis of the solutions found during the search. The height of the bar is the mean, the black bars is ± 1 standard deviation.

7.3.1.5 Number of iterations required to reach the maximum value

The iteration when the maximum score is found reveals whether the search has navigated the chemical space efficiently. The fewer the iterations required to converge the search upon its maximum score, the better. Figure 7-18 shows the iteration when the maximum value of the search was found, and no consistent trend is seen due to the large standard deviations across all problems. However, for ablation of simulation for the alogliptin and aripiprazole problems the searches show a sharp reduction in number of iterations required to find the maximum score. The opposite is seen for ambrisentan, this would suggest that in the algoliptin and aripiprazole problems the search finds the best solution early due to high scoring intermediates existing along the path. Whereas in the ambrisentan example the tree explored more due to lower scoring intermediates and no clear path being available and thus used more iterations to find the maximum score. Finally, for ablation of selection for ambrisentan did not improve the maximum score from the first expansion from the root node.

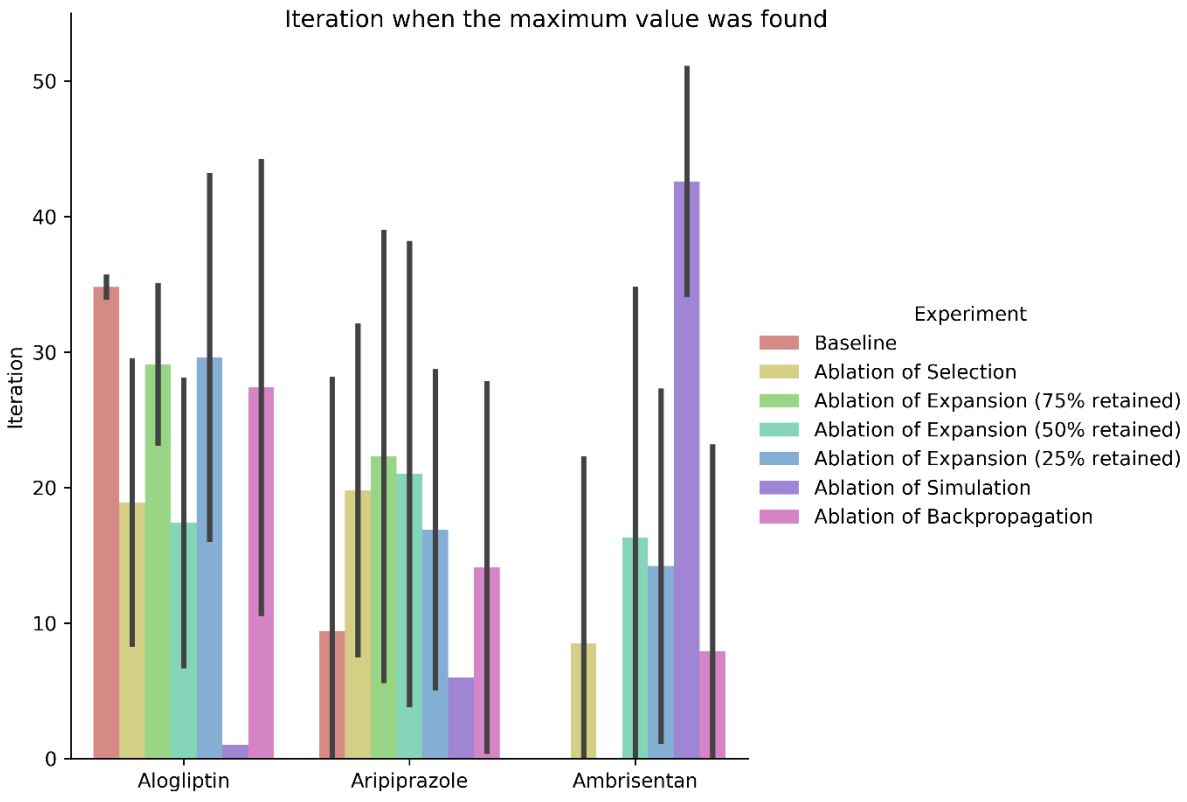


Figure 7-18 Iteration when the maximum value of the search is found. The black bars represent ± 1 standard deviation.

7.3.1.6 Depth

The maximum depth reached by the tree is an essential property to solve rediscovery problems consisting of multiple synthetic steps. For example, to solve the ambrisentan problem, the maximum depth reached must be greater than or equal to four synthetic steps. Figure 7-19 shows that in almost all cases, the maximum depth reached is three; therefore, none of the ambrisentan experiments can find the goal compound due to insufficient depth. The ablation of simulation did grow trees of greater depth. This would suggest that the ablation of simulation is greedier and is forced down the same path multiple times. Interestingly, in the case of ablation of expansion (with 25% retained), the trees are equivalent in depth to ablation of simulation. However, this increased depth does not yield better performance due to the removal of key intermediates.

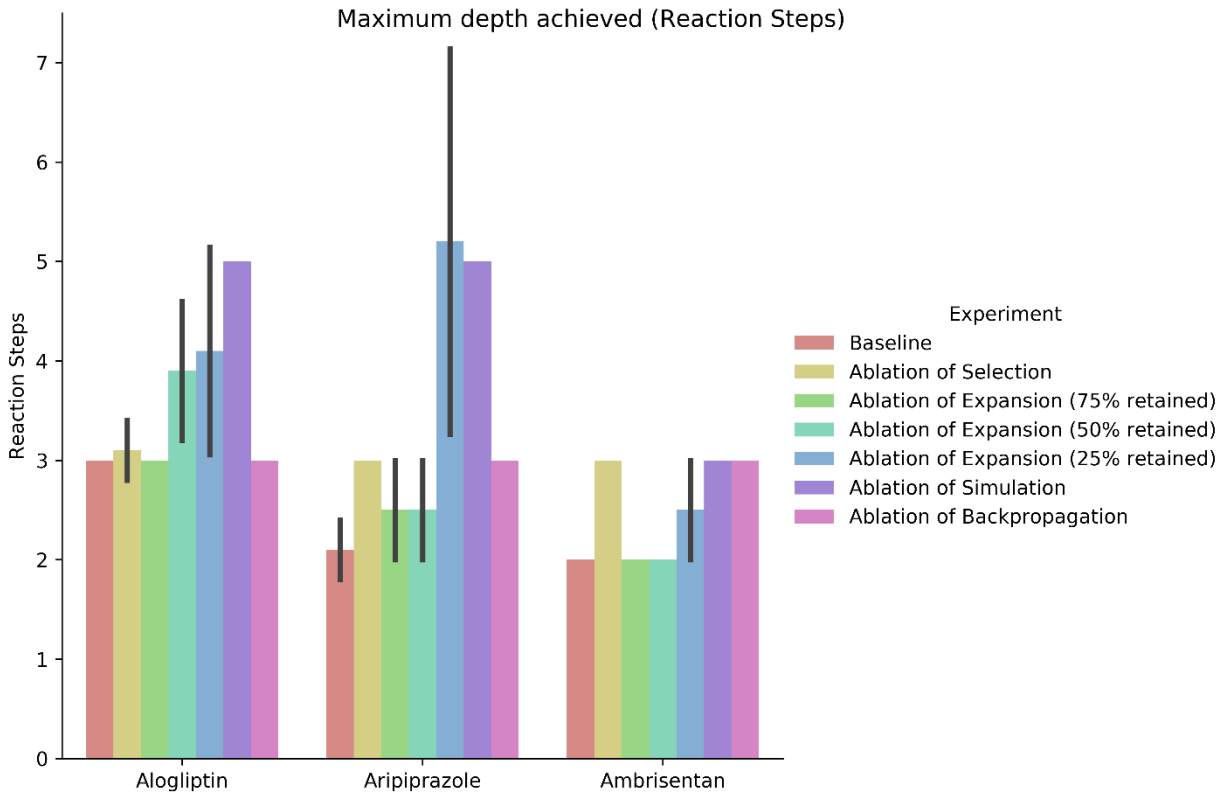


Figure 7-19 Maximum depth achieved by the tree during the search. The height of the bar is the mean, the black bars is ± 1 standard deviation.

7.3.1.7 Diversity

The pairwise Tanimoto similarity was calculated for every pair of valid molecules generated and is a measure of the diversity in the set. High values of average pairwise similarities would suggest that many of the molecules are structurally similar, and therefore the search has focused on regions of the search space with structurally similar motifs. In contrast, lower average pairwise similarities would suggest more exhaustive exploration. Figure 7-20 shows the pairwise similarities for the entire trees for the three problems.

All methods, except for ablation of simulation, performed the same as the baseline. The average pairwise similarities of around 0.2 reflect the lack of focus in one region of the search space. However, across all problems, ablation of the simulation produced average pairwise similarities higher than the baseline; this suggests that ablation of simulation caused the search to move to one region of the search space and consistently generated molecules with similar structural motifs. This further suggests that the ablation of simulation yielded a greedy search that exploited more than it explored.

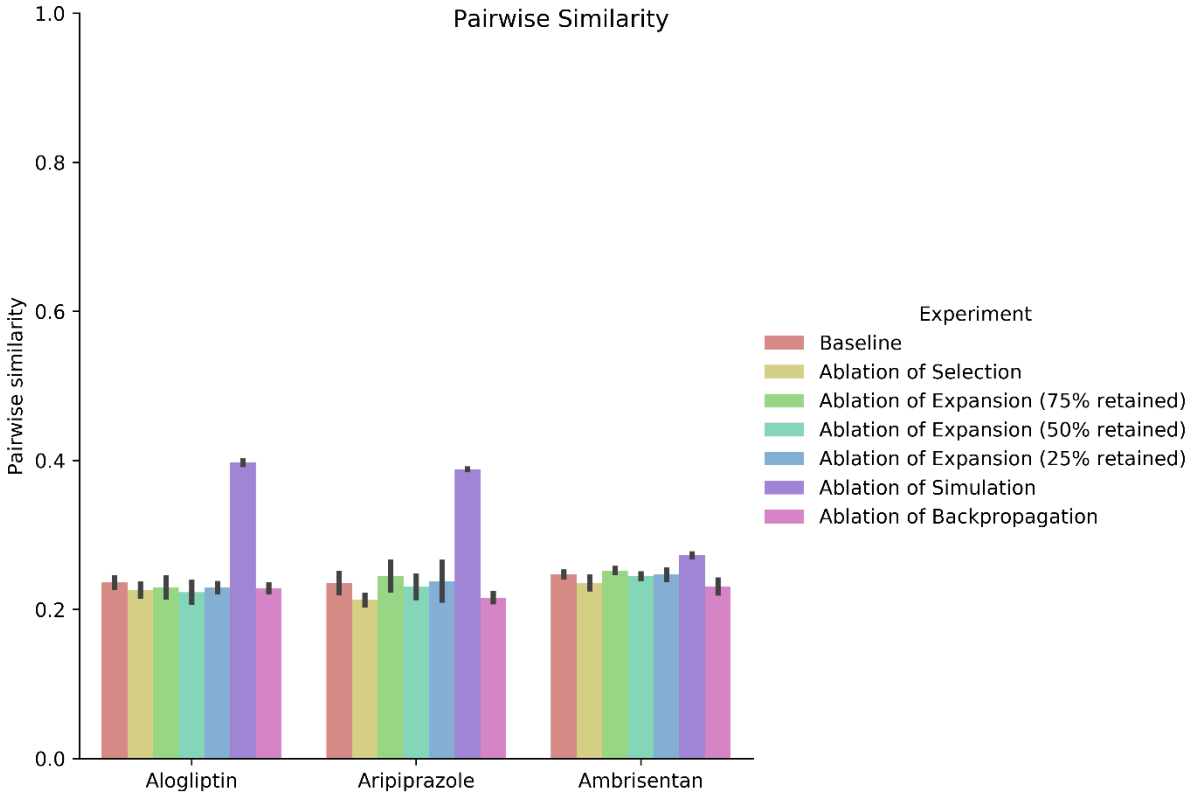


Figure 7-20 Average pairwise similarity for the tree. Black bars represent ± 1 standard deviation

7.3.1.8 Tree size

An essential aspect of the computational efficiency required to run the MCTS is the size of the generated trees. Larger trees consisting of more molecules lead to more simulations, and therefore more time is needed to complete the search, and more memory is required. Figure 7-21 shows the number of molecules generated in the tree. The figure shows that the ambrisentan problem generated more molecules than the other two problems, and this was consistent across all experiments. Arguably for alogliptin and aripiprazole, the majority of experiments yielded similar sized trees. Comparing the generated trees, the number of molecules generated for ablated selection, ablated simulation and ablated backpropagation were equivalent and significantly larger than the baseline. The explanation for this can be that when the node to expand next is random, as is the case for ablation of selection and backpropagation, fewer iterations are wasted on molecules that cannot be expanded. The same result can be inferred for the ablation of simulation, which also avoids the dead ends by directing the search to a different region compared to the baseline.

When comparing the total number of molecules generated, including both the tree and the simulation, the outcome was different. As seen in Figure 7-22, the number of molecules generated increased from alogliptin to aripiprazole and then to ambrisentan, and this is consistent with the timings in Figure 7-24. For ablation of both selection and backpropagation, the searches generated a huge number of molecules with the total above 3 million. Interestingly the standard deviation of the total number of molecules generated is the highest for ablation of selection and backpropagation. This increase in standard deviation is most likely due to the random nature of the search. Finally, ablation of simulation yielded a comparatively tiny number of molecules, as in this case, only the tree is generated. Thus, while the total number of molecules generated by ablated simulation is less than the baseline, the performance is equivalent to that of the baseline and frequently better than methods that generate more molecules.

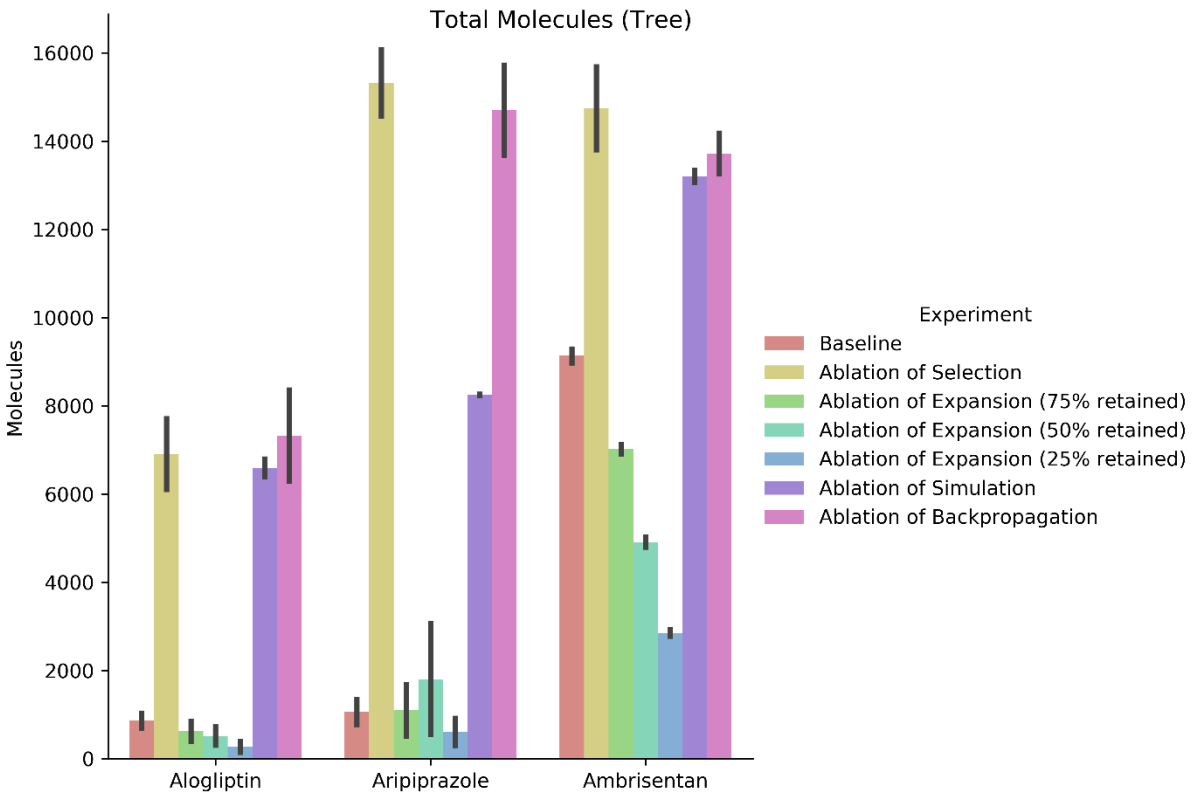


Figure 7-21 Total molecules generated in the tree. The height of the bar is the mean, the black bars is ± 1 standard deviation.

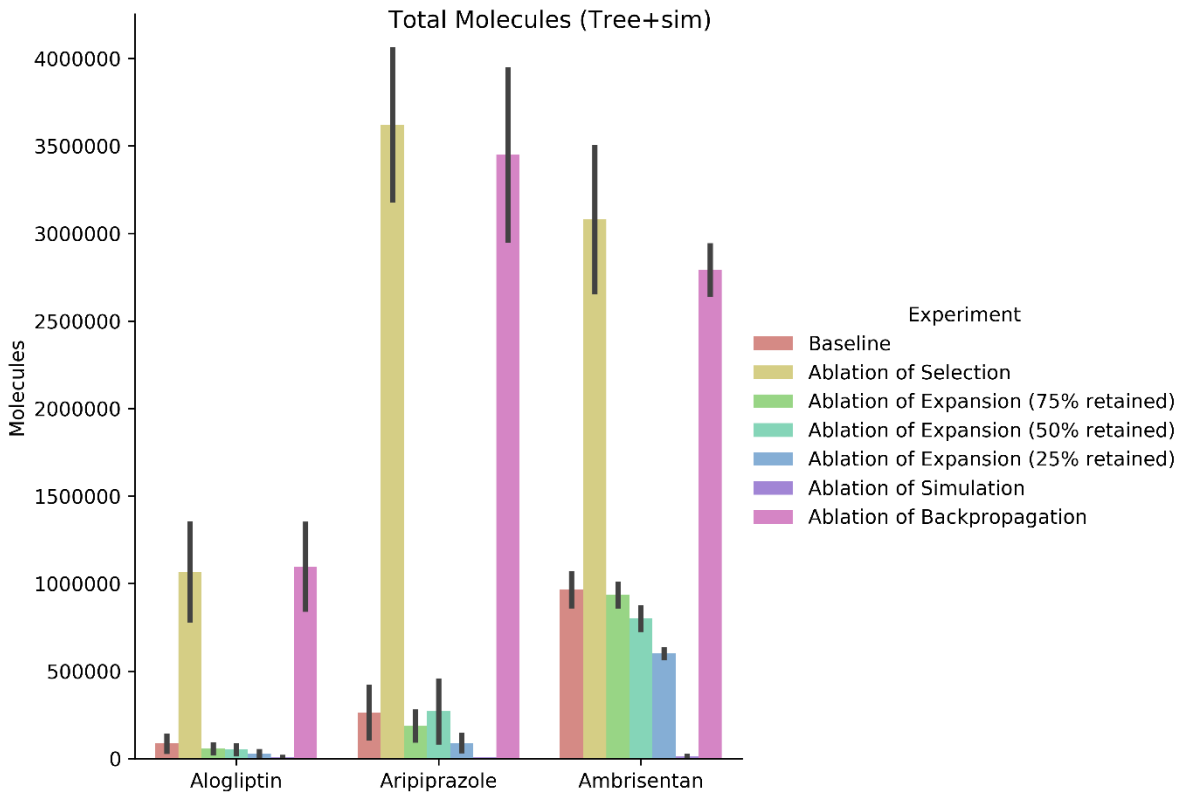


Figure 7-22 The total molecules generated during the search. The height of the bar is the mean, the black bars is ± 1 standard deviation.

7.3.1.9 Branching factor

The branching factor is a measure of the number of child nodes per parent node. Higher values of the branching factor will generally lead to slower searches and poorer performance for the RVMCTS due to the large number of decisions to choose from. A branching factor is also a valuable tool for understanding the complexity of search spaces. Figure 7-23 shows the branching factor for each experiment. Considering first the baseline, the branching factors for aripiprazole and ambrisentan were significantly larger than for alogliptin and are likely due to the larger molecules being generated in the search tree. Larger molecules will contain more reaction centres, and thus more RVs will apply to each molecule generated, thus leading to the observed higher branching factor. This would suggest that the difficulty of both ambrisentan and aripiprazole are roughly equivalent. Ablation of selection and ablation of backpropagation yielded equivalent high branching factors for all searches. Both these ablation experiments resulted in random searches, and the high branching factors suggest that the trees grew wide but not deep. Ablation of expansion resulted in a progressive decrease in the branching factor as the number of nodes retained was reduced, which is expected. Finally, the ablation of simulation is interesting as the branching factor generated during the search was high and is likely due to the larger molecules occurring at deeper depths in the search tree leading to an increase in the number of reaction centres and thus the higher branching factor. Moreover, for the aripiprazole problem, ablation of simulation yielded a similar branching factor to the baseline, and therefore it is possible that in the aripiprazole experiment, ablation of simulation reached dead-end unexpandable molecules.

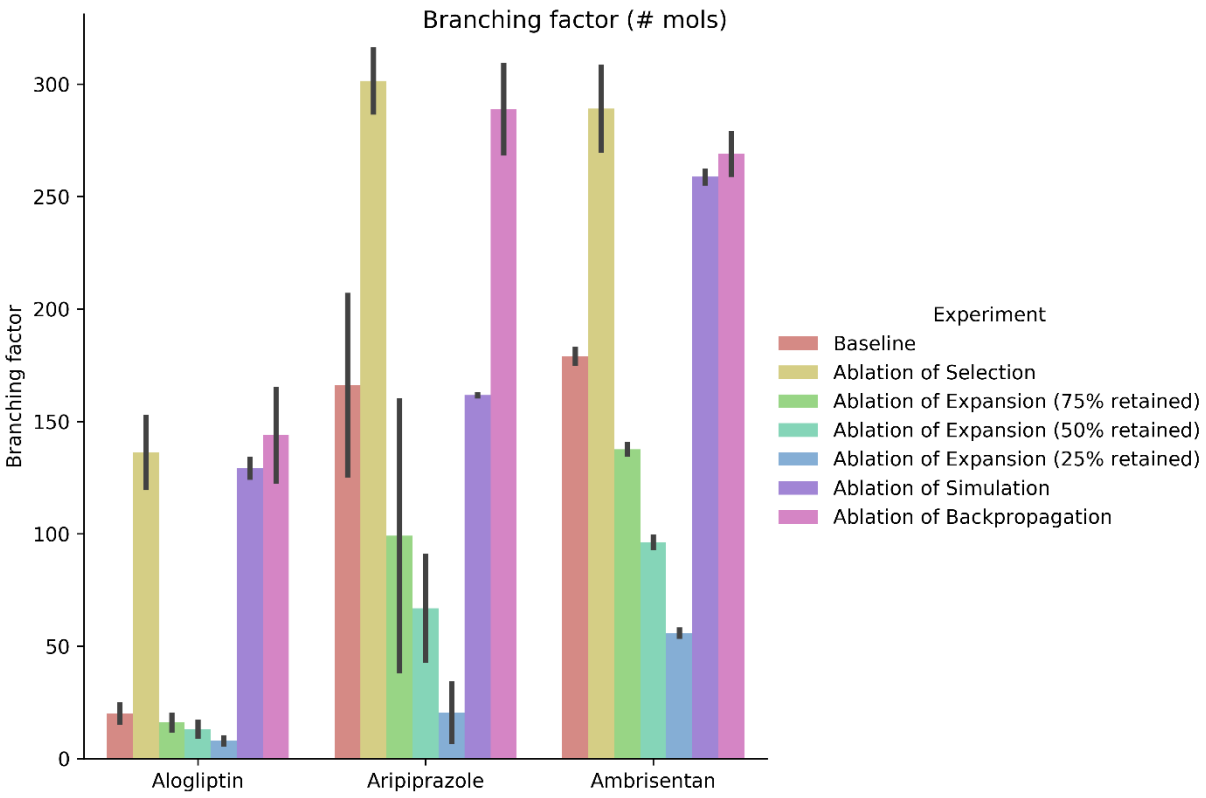


Figure 7-23 Branching factor of the tree. The height of the bars represents the arithmetic mean of the replicates. The black bars represent ± 1 standard deviation of the data.

7.3.2 Resources used

7.3.2.1 Computational cost (Time)

Computational cost is an important resource to consider when investigating the efficiency of an algorithm. It is important to note that the computational cost will differ depending on what device the algorithm is running on. Therefore, the computational cost must be measured relative to the baseline run on the same machine and approximately the same computational load (algorithm only).

The time taken for the searches to complete is shown in Figure 7-24. The times for the baseline experiments increase across each problem with alogliptin taking the least amount of time and ambrisentan the most. The increase in time taken can be attributed to the increase in the number of generated molecules. Generating molecules is the most computational expensive process in the entire search. The molecule generation process is computationally intensive due to the database transactions and the structural transformations required when applying a RV. In contrast, other parts of the MCTS algorithm, such as selection and backpropagation, are fast due to them being updating values in memory.

Ablation of selection led to a dramatic increase in the overall time taken for the searches to complete, which was surprising. As highlighted previously, the only significant increase in time can be attributed to the increase in the number of generated compounds. If more compounds are generated, then the time for completion will increase substantially due to each newly expanded node also having to be simulated. This is likely due to the removal of the UCT equation successfully avoiding poor scoring dead ends and unexpandable molecules, leading to more nodes needing to be simulated. Ablation of expansion led to a consistent decrease in the total time taken as more nodes were removed. The decrease in time was entirely due to fewer expanded nodes having to be simulated. Ablation of simulation led to the largest decrease in time take overall. This is due to the expensive computational step of simulation having been removed entirely. Ablation of backpropagation led to an increase in time taken for the same reason as the ablation of selection. As all nodes in ablation of backpropagation have the same UCT score (0), the choice amongst them is random. Random choice led to no dead-ends, and thus more molecules being generated. This leads to an increase in time taken.

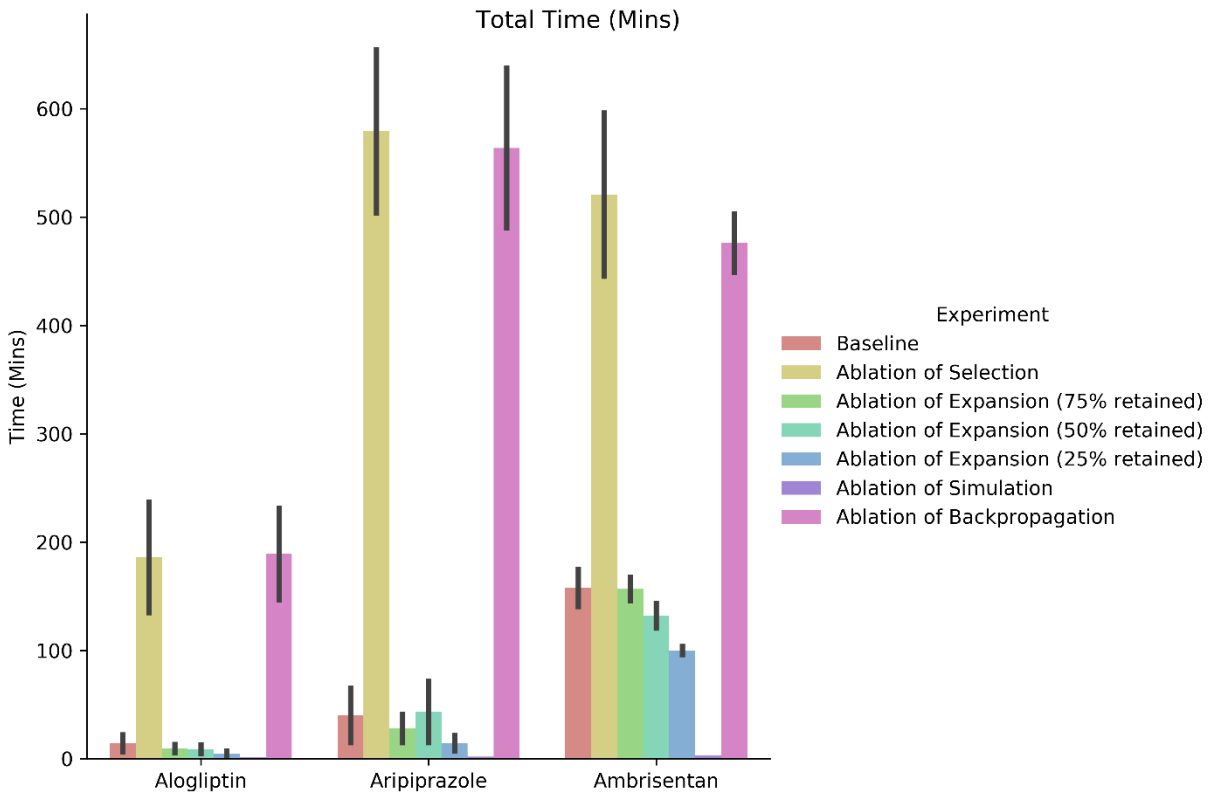


Figure 7-24 Total time taken for each replicate. The height of the bar is the mean. The black bars are ± 1 standard deviation.

7.3.3 Robustness

Investigating the robustness of an algorithm helps understand how an algorithm will perform on new problem types and new problem instances. In this investigation, three problems have been explored in the benchmark, and therefore it is hard to draw universal claims of performance. Therefore, in this analysis, two metrics will be explored, standard deviation over the maximum scores generated and the percentage of RVs used from the validated benchmark.

7.3.3.1 The standard deviation of maximum scores

The standard deviation of maximum scores helps understand whether the algorithm consistently finds the same molecule over different replicates. For example, a low standard deviation would suggest that the search has consistently found the same maximum scoring compound each time. Conversely, a high standard deviation would suggest that the search is unstable and cannot consistently find the same high-scoring state.

Figure 7-25 shows that the baseline search had zero standard deviation in the alogliptin and ambrisentan problems and an increased standard deviation in the aripiprazole problem. This would suggest that the amount of search instability is low for the base implementation of the algorithm. In contrast, ablation of selection and ablation of backpropagation yielded high standard deviations due to them being random searches and therefore can be interpreted as having high degrees of search instability. Furthermore, ablation of expansion showed that as the amount of material removed from the tree randomly increases, the amount of search instability also increases. Finally, the ablation of simulation yielded no search instability due to it being a deterministic greedy search.

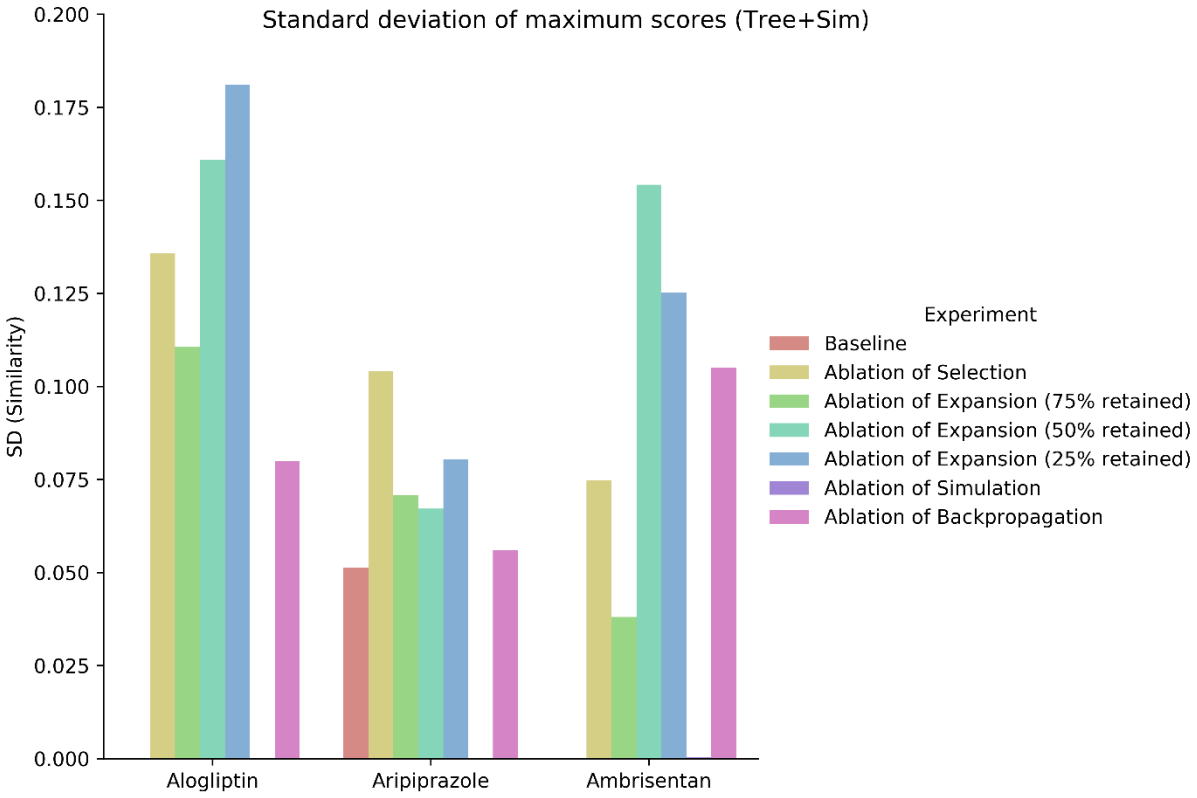


Figure 7-25 Standard deviation of maximum scores. The height of the bar is the arithmetic mean of the replicates.

7.3.3.2 Percentage of reaction vectors used from the validated database

When considering different types of trees that are being grown and whether a random search has occurred, one way to explore this is to examine the number of RVs applied as a percentage of the entire database. As the number of iterations is fixed, an increase in the percentage of unique RVs that are applicable suggests an increase in unique reaction centres, and thus more diverse molecules being generated. The percentage of RVs that were applied is shown in Figure 7-26 below, where ablation of selection and ablation of backpropagation, being random searches, yielded far more unique RVs applied compared to the baseline. Furthermore, an increase in depth yielded from ablation of simulation also meant that more unique RVs are applied.

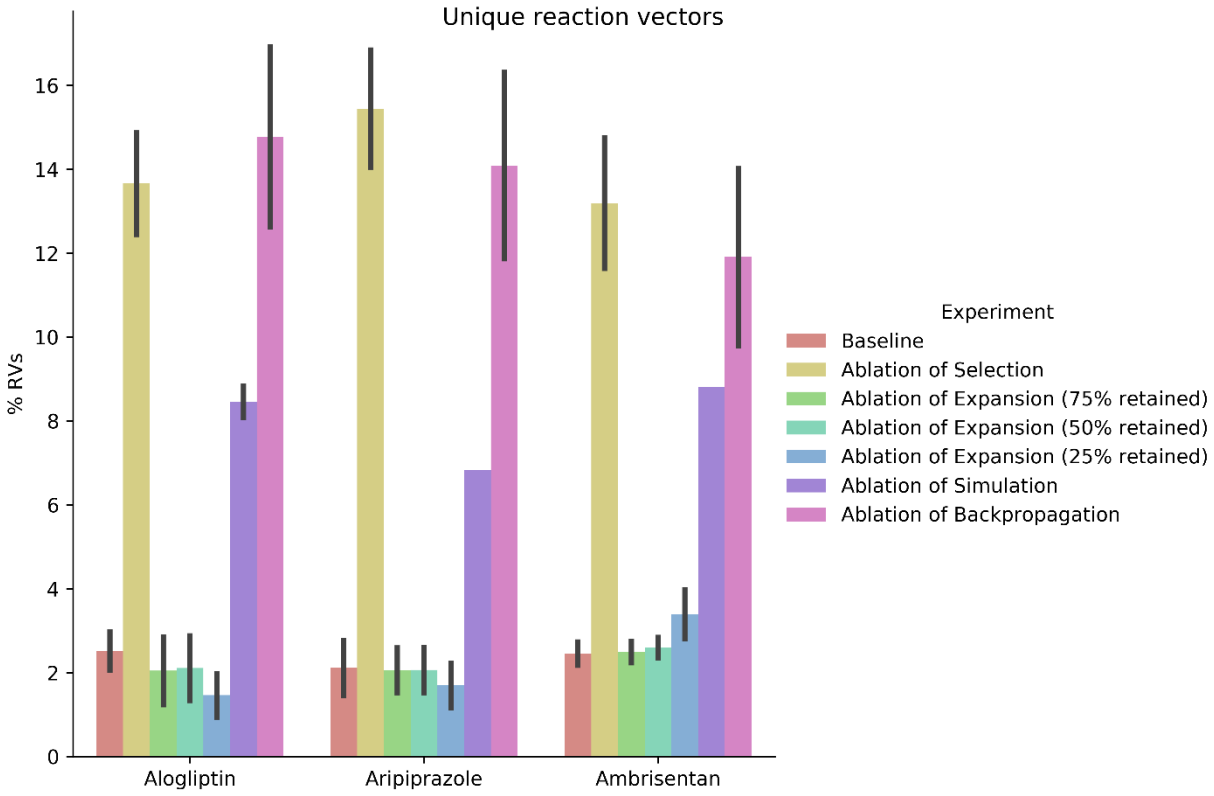


Figure 7-26 Percentage of unique RVs that have been applied during the expansion phase. The height of the bar is the mean, the black bars is ± 1 standard deviation.

7.3.4 Qualitative assessment of Algorithm performance

When investigating an algorithm, there are often qualitative aspects of the algorithmic performance that can provide additional insights into the approach's effectiveness. The qualitative assessment focuses on aspects of the search which cannot be described effectively via numeric values. In this qualitative assessment of algorithm performance, three distinct aspects are explored; visualisation of the trees, maps of the explored chemical space, and example molecules that have been generated. The maps of the chemical space explore how different searches explored different regions of the space. Finally, example molecules are drawn from the pool of generated molecules and include both high scoring and poor scoring molecules.

As 210 searches (10 replicates, three problems, seven different configurations) were run, it is impractical to visualise all possible searches. Hence, 21 (1 from each problem and configuration) are selected at random. The tree visualisations are shown only for the aripiprazole problem. Furthermore, only one example of ablation of expansion (retained 25%) is depicted. The remaining tree visualisations from the 21 randomly selected searches are shown in appendix D.

7.3.4.1 Tree visualisation

Each tree is ordered in the following manner. The nodes on each level are sorted by their corresponding number of visits. The thickness of each line from the parent node to the child node is determined by the number of visits to the corresponding child node. The tree layout is computed using the EoN (Epidemics On Networks) package(Miller & Ting, 2019). The nodes are coloured by their similarity score to the target molecule with a value of one being yellow and a value of 0 being dark purple. The baseline search depicted below shows four levels of the tree; in total, the number of expansions that occurred is low (5, including the root node). The low number of expansions suggests that the search is being trapped in a local optimum, which means that most iterations are being wasted on returning to the same dead-end node. Ablation of selection shows a substantially larger number of expansions compared to the baseline. Ablation of selection resulted in a random search, as highlighted from the quantitative analysis of the tree. However, as there are a large number of molecules at the first level of the tree the depth reached by the ablated selection tree is limited. Ablation of expansion shows a tree with more expanded molecules than the baseline. This suggests that ablated expansion has removed some dead-end molecules leading to more compounds being expanded. However, the ablation of expansion is a random removal of nodes and key intermediate molecules were also removed, and thus the overall performance of the search was not improved. Ablation of expansion (25% retained) still suffered from many nodes being present in the first layer of the tree and the tree did not grow deeper. This search was randomly chosen from the replicates in which the mean depth of ablation of expansion (25% retained) was depth five,

therefore, this example is an outlier. Ablation of simulation generated a tree substantially more asymmetrical with a deep path down the right-hand side of the tree compared to the baseline. The depth of the tree is greater than that of the corresponding baseline and has higher scores that are observed at the bottom of the tree as well. Finally, the line thickness is the same throughout all edges in the ablation of backpropagation search as no visit count information was recorded. The shape and overall structure of the ablation of backpropagation tree are the same as the ablation of selection tree as they are both random searches.

7.3.4.2 Tree search (baseline)

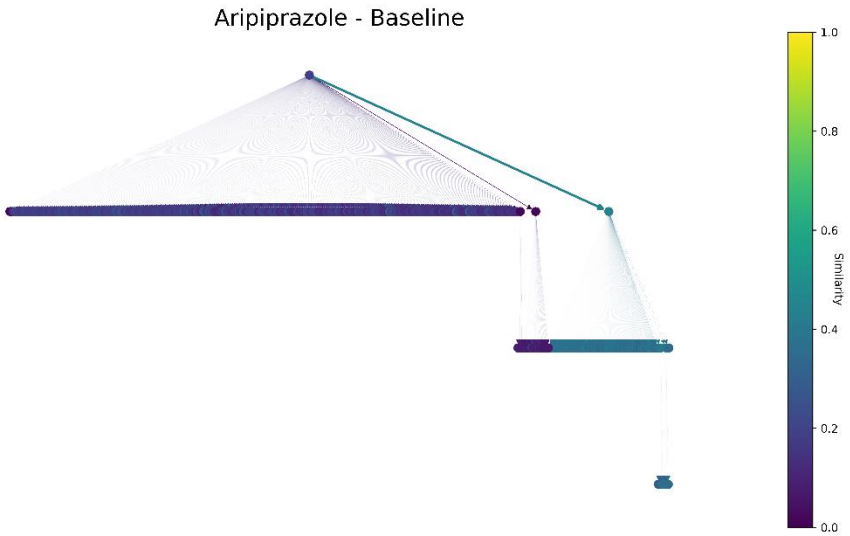


Figure 7-27 Tree visualisation of aripiprazole baseline search.

7.3.4.3 Tree search (ablation of selection)

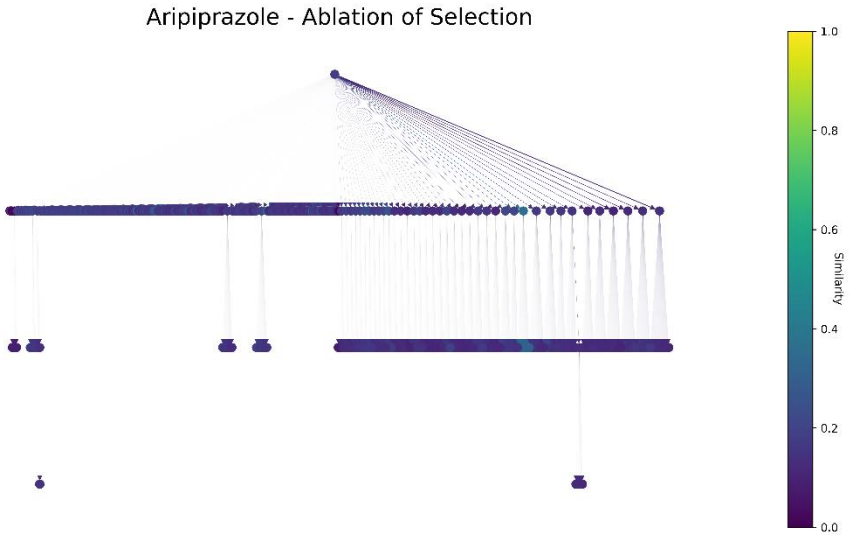


Figure 7-28 Tree visualisation of the aripiprazole ablation of selection

7.3.4.4 Tree search (ablation of expansion)

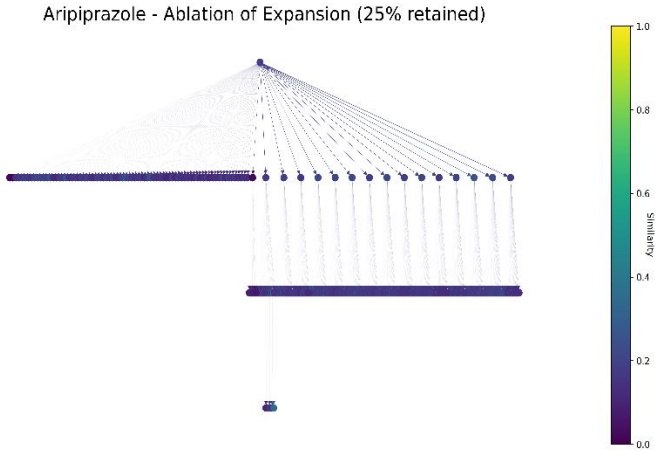


Figure 7-29 Tree visualisation of the aripiprazole ablation of expansion (25% retained)

7.3.4.5 Tree search (ablation of simulation)

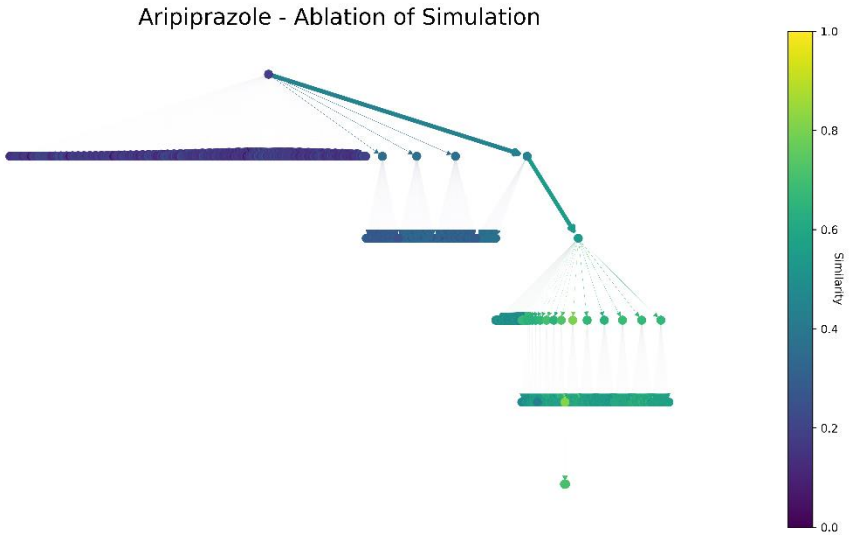


Figure 7-30 Tree visualisation of the aripiprazole ablation of simulation

7.3.4.6 Tree search (ablation of backpropagation)

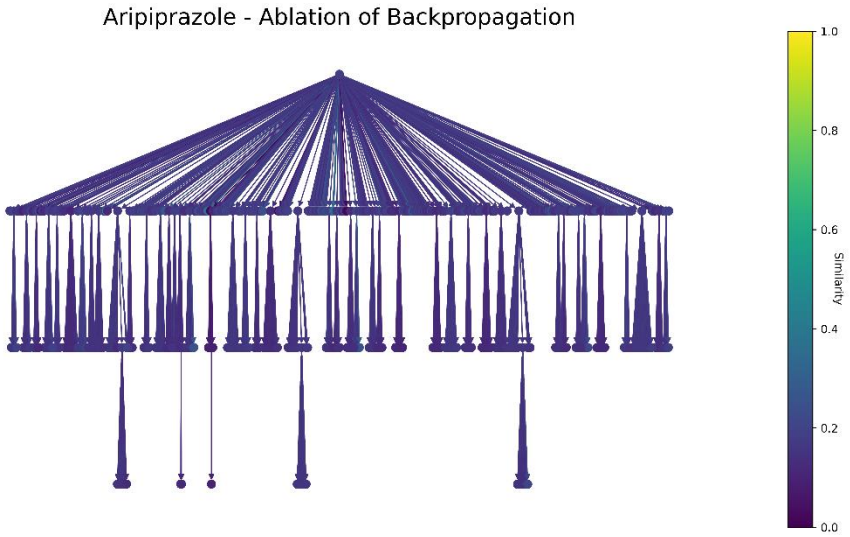


Figure 7-31 Tree visualisation of the aripiprazole ablation of backpropagation

7.3.4.7 Maps of generated chemical space

Figure 7-32, Figure 7-33, Figure 7-34 below are maps of the chemical space generated for each of the tree searches. Each figure shows only the molecules generated for that particular problem. The chemical space maps are generated relative to themselves and therefore only represent the space that the tree search visits, not the whole chemical space. It is important to note, the distance between the points does not infer how far away a given point is from another, just that the cosine distance of the feature vector of a point is greater than 0.15 away. Finally, points can exist beneath other points; however, this effect is mitigated mainly due to the stochastic nature of how the PCA/UMAP dimensionality reduction occurs.

First, the map of chemical space for the alogliptin problem shows that the ablation of selection (ochre), ablation of simulation (purple) and ablation of backpropagation(pink) experiments generated unique regions of chemical space. The unique regions of chemical space can be identified by the large clusters of points in a single colour. Ablation of backpropagation has spread across almost the entirety of the generated search space. This would further confirm that the ablation of backpropagation is a random search due to widespread coverage (exploration) and lack of focus on a single region. In contrast, ablation of simulation has largely focussed on two regions of chemical space. The two clusters align with the most visited paths in the tree which splits into two deep subtrees at depth one as shown in the tree visualisation in appendix D. The ablation of expansion experiments generated some singletons (dissimilar to all other molecules), which are visible as single points on the periphery of the visualisation. The baseline is not directly visible as other searches have generated the same molecules.

Second, the map of generated chemical space for the aripiprazole has a different structure than the alogliptin problem. Again, the ablation of backpropagation and ablation of selection results in molecules spread over the generated space. In contrast, the ablation of simulation generated many unique singletons. The singletons can be seen around the periphery of the visualisation. Many singletons are due to the deeper search tree having generated molecules that are larger and more structurally diverse than the molecules generated at the shallower depths of the other searches. Finally, ablation of expansion generated some unique singletons, which can be seen on the edge of the visualisation; the baseline is again not visible due to the other searches having generated the same molecules.

Finally, the generated map of ambrisentan shows the ablation of backpropagation spreading out over the entire search space. This would suggest that all experiments performed poorly as no clusters are observed except the large central mass. On the other hand, the large number of

singletons visible would suggest that substantial exploration did occur, however, this was unsuccessful in finding a good solution.

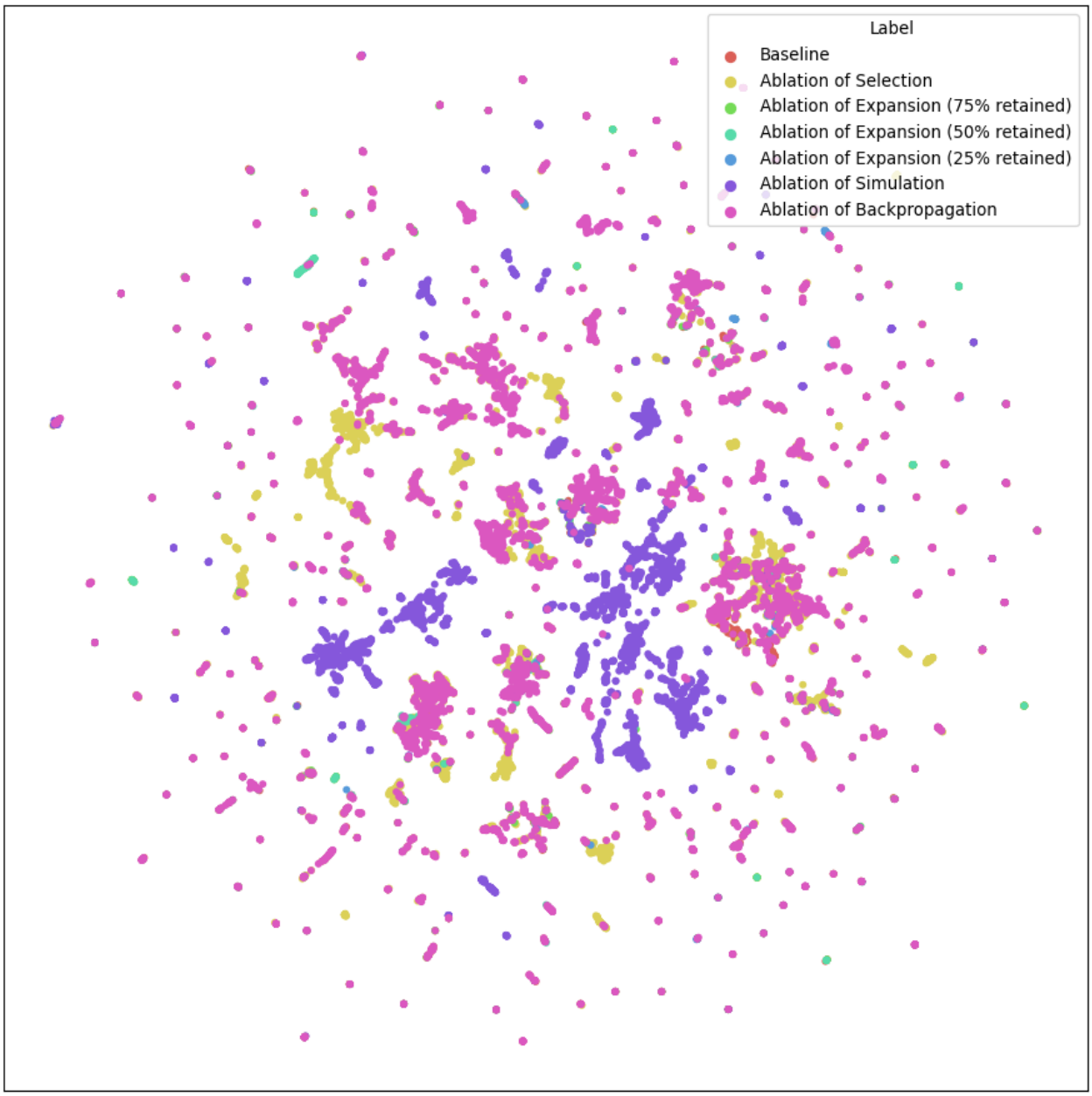


Figure 7-32 Chemical space of the generated compounds for the alogliptin problem. The searches chosen have been chosen at random.

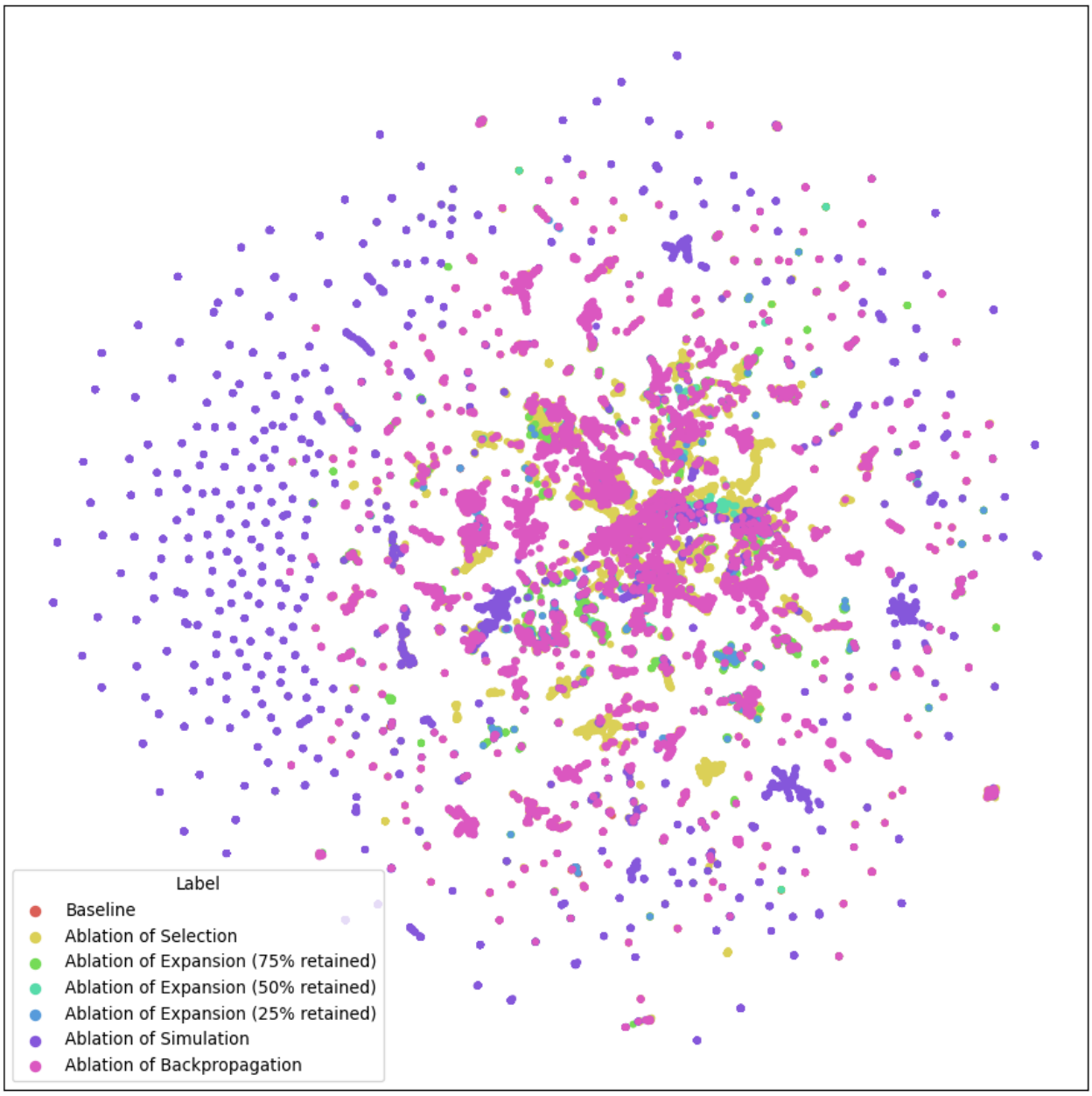


Figure 7-33 Chemical space of the generated compounds for the aripiprazole problem. The searches chosen have been chosen at random.

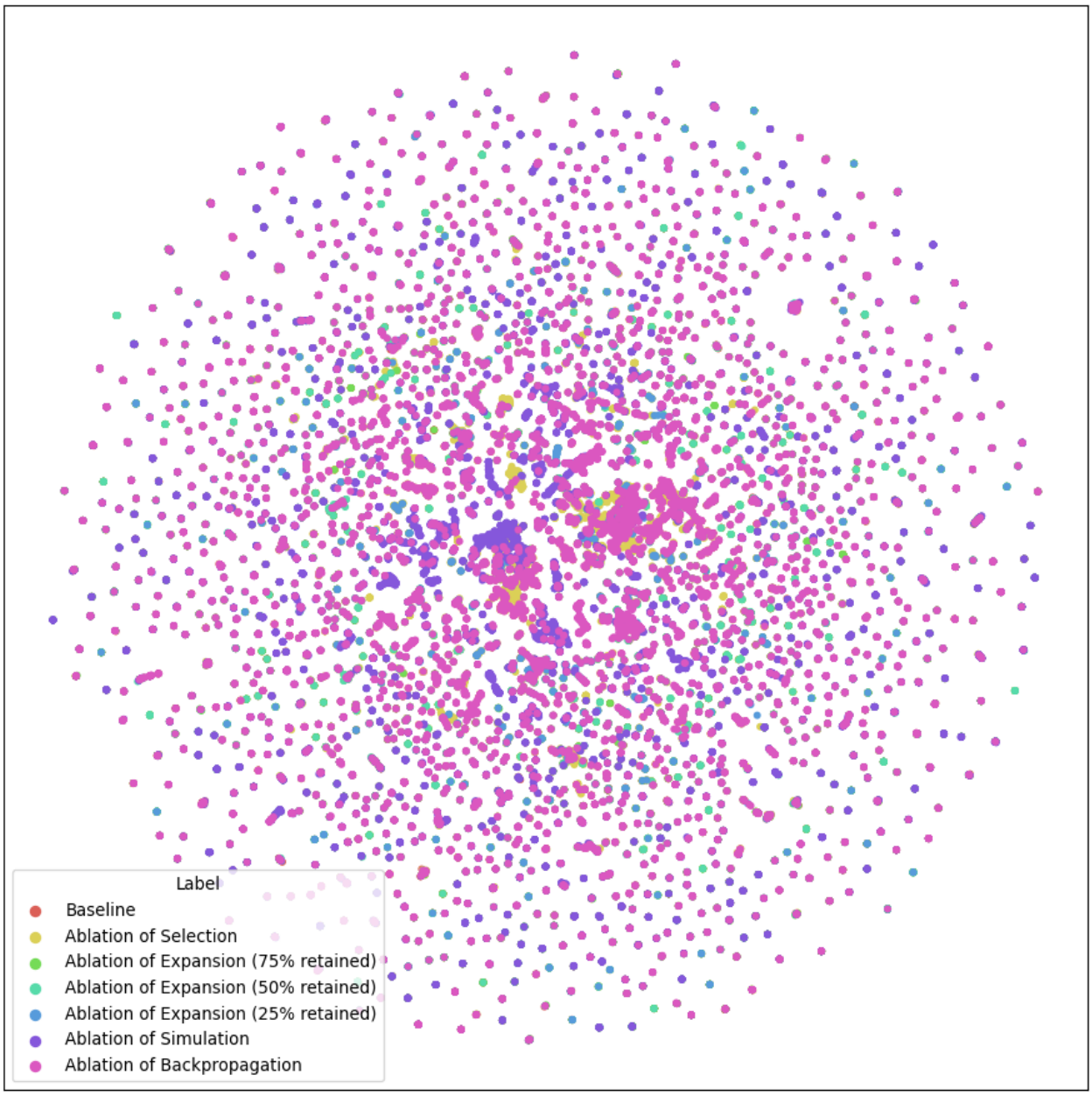
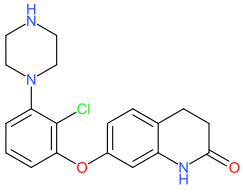
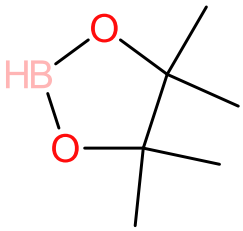
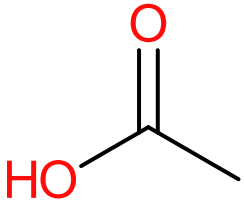


Figure 7-34 Chemical space of the generated compounds for the ambrisentan problem. The searches chosen have been chosen at random.

7.3.4.8 Molecule examples

In the analysis of a de novo design algorithm, compounds must be visually inspected. For some of the ablation experiments carried out in this chapter, the total number of molecules generated was greater than three million and so it is unfeasible to visualise all compounds. Therefore, a few compounds were selected from each of the ablation studies. These are: best compound generated; worst compound generated (excluding invalid molecules); and worst compound generated with the most number of visits (excluding invalid molecules). Again, only the compounds generated for the aripiprazole problem are shown in Table 7-6. Ablation of backpropagation does not have the minimum most visited compound due to no visit counts being updated in the tree. The rest of the compounds are shown in appendix E.

The compounds generated show several interesting trends. First, all the compounds with maximum scores for aripiprazole contain several fragment substructures present in aripiprazole. This is promising as it suggests high scoring compounds can be generated; however, they are being lost amongst poorer scoring compounds. Second, the minimum scoring compounds are all by-products from RVs. These by-products are generated during the application of a RV which yields a pair of compounds. These compounds should be removed from the tree as they are not helpful and are not the intended product from a given RV. Third, the minimum most visited compounds are compounds that score poorly yet have been visited many times; these compounds are dead-end compounds. These compounds generally do not yield paths to high scoring regions of chemical space, yet the search has spent many visits exploring them. An example of a dead-end is the acetic acid (example: baseline minimum most visited). Acetic acid was visited several times during the search yet provided no benefit to the overall search. Moreover, any compounds generated from acetic acid would have been synthetically intractable. Interestingly, in the case of the ablation of simulation, the worst scoring but most visited compound was an intermediate that was chosen, which led to the highest-scoring compound during the search.

Experiment	Maximum	Minimum	Minimum most visited
Baseline			

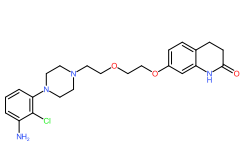
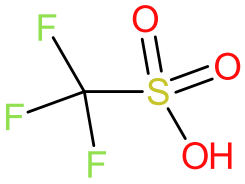
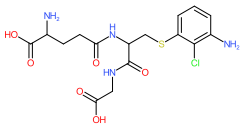
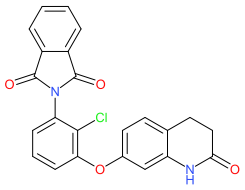
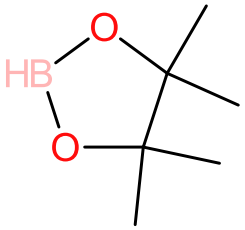
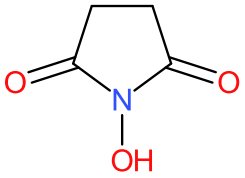
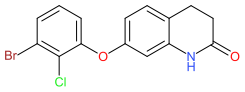
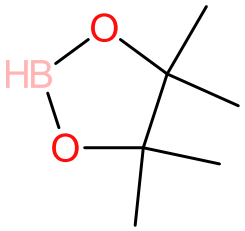
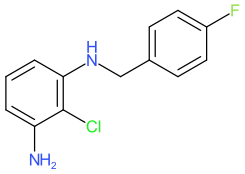
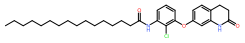
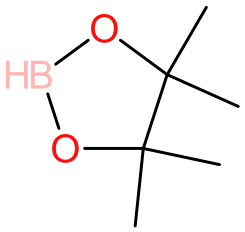
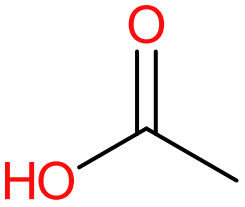
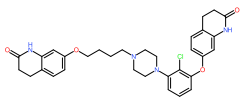
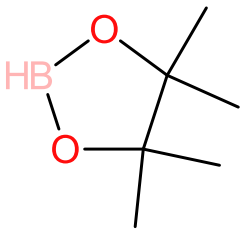
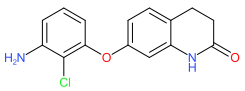
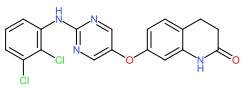
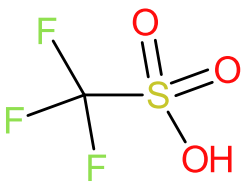
Ablation of Selection			
Ablation of Expansion (75% retained)			
Ablation of Expansion (50% retained)			
Ablation of Expansion (25% retained)			
Ablation of Simulation			
Ablation of Backpropagation			Cannot be calculated due to no visit counts being available.

Table 7-6 Example molecules found during the replicate search of aripiprazole. The examples are the maximum scoring molecule, minimum scoring molecule, and the minimum scoring molecule normalised by visit count.

7.4 Conclusions

In this chapter, several ablation experiments were carried out and the results were compared to a fully operational RVMCTS. The problem type was the rediscovery problem, based on the syntheses of three known drugs. These were hand encoded from the literature with the required RVs added to a database of RVs. The in-silico reactions were validated using the RV engine. The ablated experiments were analysed using three separate criteria: algorithmic performance, computational resources, and robustness.

Several trends were observed in algorithmic performance. First, the baseline RVMCTS had promising performance across alogliptin and aripiprazole. Second, the ablation of selection and ablation of backpropagation yielded comparable algorithmic performance due to the search reducing to a random search. The greater degree of stochasticity in these random searches led to, in some cases, better performance than the baseline. Third, ablation of expansion led to an increase in search instability but also an increase in depth. Finally, the ablation of simulation outperformed the baseline search due to the ablation of simulation being greedier and growing into deeper regions of the search space. Analysis of computational resources used demonstrated that the full enumeration at expansion and the simulation were the most computationally expensive steps in terms of both time and the number of molecules generated. Finally, investigations into the robustness of the RVMCTS revealed that the removal of nodes during the expansion phase led to the highest increase in search instability.

Overall, this leads to several key conclusions. First, a rational method of node selection needs to be used to prioritise nodes in the tree, as without this, the search degrades to random. Second, the pruning of nodes from the tree speeds up the search. However, there is the risk of removing nodes that are required to reach the end goal state, which can therefore lead to degraded performance. Third, removal of the simulation stage does not reduce maximum search performance and decreased the overall computational expense of the search. However, removal of simulation, made the search easier to deceive. Finally, ablation of backpropagation yielded the performance of a random search, and therefore should be maintained. The next chapter further explores the simulation module and its influences on the search.

Chapter 8 Investigations into Simulation

8.1 Introduction

Chapter 7 demonstrated that the RVMCTS was a promising algorithm for de novo design. An essential aspect of the basic MCTS is that scoring occurs via a stochastic depth-first search to terminal criteria. This lookahead score is vital as it allows the node to see past local optima. A key feature highlighted during the ablation studies in Chapter 7 was the overall computational cost of simulation. The computational cost of simulation in the search of the RVMCTS is due to it being implemented as an exhaustive enumeration. Therefore, the computational cost of simulation scales poorly with a large number of nodes in the tree. Hence, in this chapter, the focus will be on exploring simulation further and understanding how the computational cost of simulation can be reduced. First, this chapter introduces the basic simulation algorithm and the motivation for using the simulation stage during the search. Next, the chapter explores different implementations of a depth-first stochastic simulation in the RVMCTS. The final section focuses on the experimental exploration of a machine learning-based simulation in contrast to a stochastic search.

8.2 Simulation: the basic algorithm

As discussed in Chapter 6 and summarised here, the basic implementation of the MCTS requires the use of a random lookahead. The random lookahead is where the search continues until a terminal end criterion is reached is the end of a sequence of actions. For example, in chess, it would be the end of the game (win, loss, draw) whereas in reaction-based de novo design, it is the molecule at the end of a reaction sequence. The end state is then converted to a number using an objective score which is backpropagated up the tree. In the basic MCTS algorithm, three criteria are needed to perform a simulation: an objective function; the ability to choose an action; and the terminal criterion. As highlighted in Chapter 6, the basic algorithm uses random (Monte Carlo) action selection during simulation to choose actions.

In the implementation of the RVMCTS described in Chapter 6, the simulation definitions are the following: the objective function is the similarity to a goal molecule; the terminal criterion is the depth one full enumeration from the expanded node; and the ability to choose a random action is not implemented as a full enumeration is performed. This implementation leads to several issues for the RVMCTS. First, a full enumeration for a given molecule is expensive to compute because it requires a scan of all applicable RVs, then for each applicable RV, a full scan of the reagent pool is required, and finally, all applicable RVs and corresponding reagents are applied. As all nodes generated during expansion need to be simulated using an enumeration to depth one, this leads to a

massive computational expense. Second, the terminal criterion of the simulation should represent when the search ends, for example, in chess, checkmate is the end of the game. In the RVMCTS, however, a complete enumeration to depth one does not represent the endpoint of the search. Third, by using a depth one full enumeration, the ability of the search to avoid being trapped in a local minimum is substantially reduced since this requires the horizon of the search to be extended beyond a single action.

In this chapter, different options for simulation are explored in the RVMCTS. The objective function is constant for all the searches and is predefined before the search begins, however, different implementations of the terminal criterion and the action generation process are explored.

8.3 Methods

This section describes the implementation of the terminal criterion and random action selection.

8.3.1 Terminal criteria

The terminal criterion dictates when the search will end and is based on the current state or node properties. For example, in the RVMCTS, a terminal criterion is defined based on the molecule and node properties. An important consideration is that the terminal criterion for simulation should reflect the final state in the search, which in reaction-based de novo design is the end product of a reaction sequence. Thus, if the terminal criterion for tree growth is depth three, i.e., a reaction sequence with three steps, this would ideally be true for both the simulation and the tree itself, that is, states in the search should not be generated beyond depth three.

In de novo design, however, terminal criteria are hard to define since the definition of when a reaction sequence is complete is often not explicitly defined. Furthermore, scoring in de novo design suffers from two problems: incomplete information and is unbounded. That is scoring functions do not capture perfectly what the final designed compounds should contain. Additionally, scoring functions are often unbounded such that scores can always be improved. In many cases, this can leave further optimisation possible even though the compounds might be "first-in-class". Finally, a terminal criterion does not need to be a single objective or property and could be more complex such as a combination of depth, physicochemical properties, and score.

In this investigation, the maximum number of reaction steps or maximum depth was used to explore simulation. For example, for aripiprazole, the maximum depth was three, and for ambrisentan, it was four.

As stated above, the same terminal criterion was used for tree growth and simulation. This allows the score from the simulation to reflect the distance from the root to the goal molecule. For

example, if the terminal criterion is depth three, any molecules generated at depth three are flagged to not expand further. Furthermore, if they are selected during the selection phase, then an empty value is backpropagated, discouraging the node from being revisited in the future.

8.3.2 Choosing a reaction vector and reagent

For a simulation, the second requirement is choosing a random action from a list of applicable actions. In the RVMCTS, selecting a random action is nontrivial due to interaction with the RV database and corresponding reagent pool. As highlighted in the literature review and summarised here, the RVs are stored in a database and therefore, to select a random RV, all applicable RVs must be found first. Following the database transaction, a single RV can then be chosen from the list. The selection of a random applicable reagent is slightly more complex. Reagents are chosen during the application of the RV, and a random reagent cannot be chosen before application. Instead, all possible reagents were applied, and a random product molecule was selected. This achieves the random choice of both RV and reagent, however, some of the computational benefits of pruning of the RVs is lost as all applicable reagents still need to be applied.

The routine for the random simulation is shown below:

1) For a given molecule, identify all applicable RVs
2) Choose one at random
3) Apply the RV with all applicable reagents and generate products
4) Choose a product at random.
5) Repeat Steps 1-4 until the terminal criterion has been reached.

This routine can be called multiple times depending on the number of desired terminal states. Finally, if the simulation did not achieve a terminal state due to steps one to four failing due to no molecules being generated, the simulation was reset and restarted from the newly expanded node. This process was attempted 100 times until the appropriate depth was reached. If this failed, then the molecule returned a score of zero.

8.3.3 Miscellaneous RVMCTS improvements

From the ablation studies tree-wide duplicate detection was implemented. Tree-wide duplicate detection keeps track of all molecules generated in the tree. If a molecule has been generated earlier in the search, the duplicate compound will not be added directly to the tree. This improvement was not tested independently and are seen as improvements for which all searches, regardless of modifications, will benefit.

8.3.4 Experimental

A small, focussed set of experiments was carried out to test the effectiveness of the simulation step. Table 8-1 Terminal depth criteria and the number of simulations. The number of simulations corresponds to the number of molecules.Table 8-1 below outlines the parameters that were tested:

Terminal Criteria	Number of Simulations
Depth 3 (Aripiprazole)	1
Depth 3 (Aripiprazole)	50
Depth 3 (Aripiprazole)	100
Depth 4 (Ambrisentan)	1
Depth 4 (Ambrisentan)	50
Depth 4 (Ambrisentan)	100

Table 8-1 Terminal depth criteria and the number of simulations. The number of simulations corresponds to the number of molecules.

Five replicates were run in this experiment, and only aripiprazole and ambrisentan problems will be explored.

8.3.4.1 Number of simulations

The number of simulations was defined based upon the number of molecules achieved at the specified depth. The number of molecules or endpoints achieved will significantly influence the accuracy of the average score that the simulation phase will generate. An average score is chosen due to the simulation being a random process such that it is unlikely that a high maximum score will be found. Therefore, the following number of simulations/endpoints were explored: 1, 50 and 100.

8.3.4.2 RVMCTS parameters

The same parameters as specified in Chapter 7 were used to allow for comparison to the baseline. These parameters are shown in Table 8-2 below. The main difference here is that the simulation type, instead of being a depth one full enumeration, is a variable that can be changed.

Parameters	Value or Setting
Total Iterations	50
Exploration Constant	0.001
Simulation type	Random (predefined terminal criterion)
Backpropagation type	Continuous

Table 8-2 RVMCTS parameters

Finally, a failsafe mechanism was implemented to prevent overly long searches. A time out of 96 hours (maximum time on the Sheffield HPC facility) was used, at which point the search was recorded as a failure, and no tree was returned.

An alternative approach to terminating the search would be to limit the total number of molecules. Rather than terminating based on time, the searches could be terminated based on the total number of generated molecules. However, this would lead to an unfair comparison as the number of iterations would no longer be constant, and thus the number of function calls (in both the expansion and simulation steps) across experiments would also not be constant.

8.3.4.3 Analysis

In Chapter 7, a complete analysis was performed to understand the effect of each component of the MCTS. In this setting, five measures were explored: solution quality through the maximum score; solution quality through the mean score; time taken for the search to complete; solution quality of the exploitation term of each node; and robustness measured by the standard deviation of maximum scores. Finally, a general, qualitative assessment is provided by visualising the trees.

8.4 Results

8.4.1 The maximum score found in the tree

The maximum score is the most important metric for determining if a rediscovery problem has been solved. Figure 8-1 below shows the maximum score found in the tree for each of the searches averaged across five replicates, with the black bar showing the corresponding one standard deviation of the maximum scores of the searches. In addition, the baseline and ablation of simulation results from the ten replicate searches carried out in the ablation studies in Chapter 7 are shown alongside the results.

The maximum similarity found across the aripiprazole problem was higher than the corresponding baseline in all cases and the random action simulation performed similarly to the ablation of simulation in Chapter 7. However, there was no discernible difference in performance across the different number of simulations. In the ambrisentan problem, all results were similar to the baseline; that is, there was no improvement in the maximum score for any if the random action simulations.

There are two possible reasons for the invariance of the results to the number of simulations. The first is that due to the large width of the search tree, it is unlikely that sufficient iterations are being carried out to see the benefits of more endpoints during simulation. However, as described below, simulation carries a considerable computational expense that would be even greater if the number of simulations was increased and would likely incur the 96-hour cut-off. The second aspect is the limited coverage of the search tree. In ambrisentan, the mean branching factor for the baseline search was approximately 175. Suppose a node is currently at depth one and needs to be expanded to depth four. This means that the possible search space beneath the node is 175^3 or approximately 4.9M endpoints. For 100 simulations, the coverage of the endpoints is 100/4.9M or 0.002% of the total search space. Hence, with such low coverage, the possibility of backpropagating a score that accurately reflects the mean score of all endpoints is low. The presence of duplicates further worsens the poor scoring.

The similar performance of the random simulations searches to the ablation of simulation would suggest that the search is being directed towards the same local optimum in both cases. Even by looking several steps past the local optima during simulation, the search still directs itself towards the local optima. This suggests that the process of simulation is not mitigating the effects of optimum trapping entirely. The focus on a single high scoring local optima suggests that the search is

still selecting greedily and is not exploring enough to visit other paths in the tree. This is in part due to the lack of differentiation in scores found during the simulation phase but also the low value of the exploration parameter in the UCT equation (C_P=0.001).

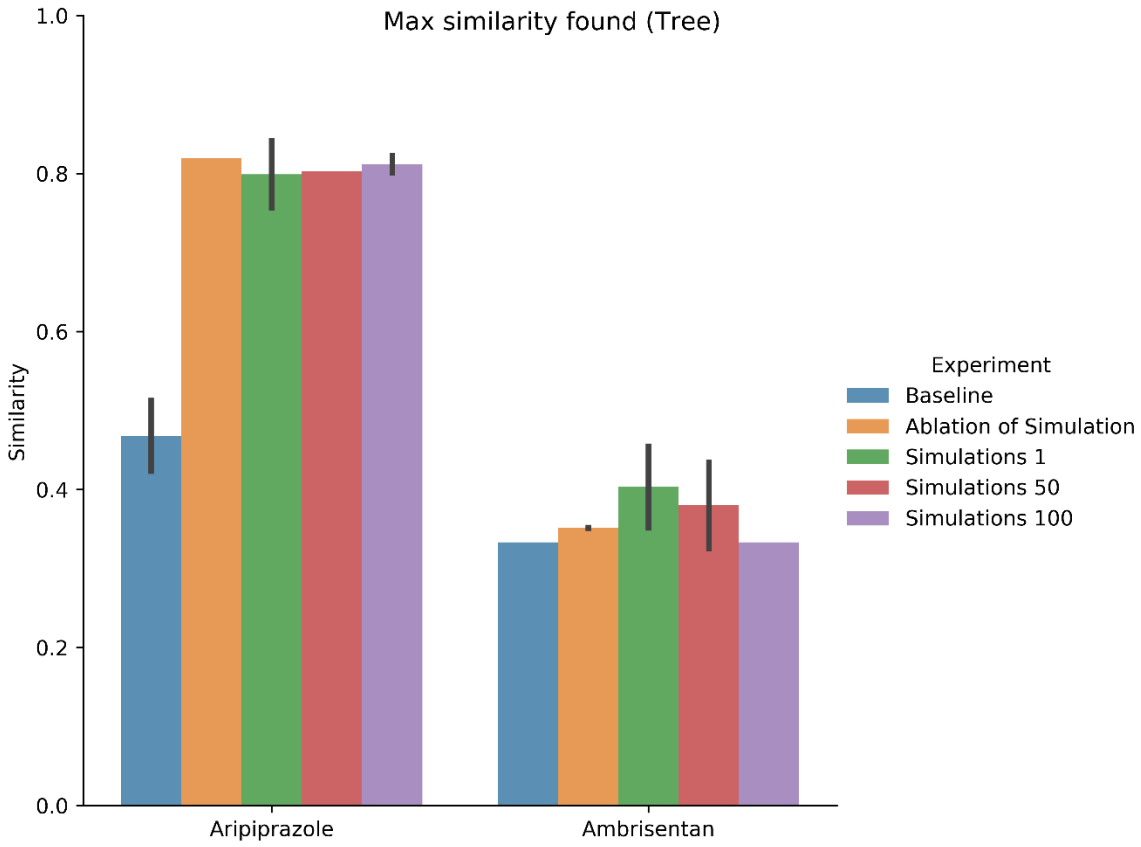


Figure 8-1 Maximum score found in the tree. The height of the bar is the mean, the black bars is ± 1 standard deviation.

8.4.2 Maximum score found in the tree and the simulation

Molecules are generated in both the tree and the simulation phase, and Figure 8-2 shows the maximum score found considering both the tree and the simulations. However, including the molecules generated during simulation did not improve the overall maximum scores. This suggests that random simulations are unlikely to find high scoring compounds due to the large number of possible compounds that could be generated at the endpoints of the sequences generated in the simulations.

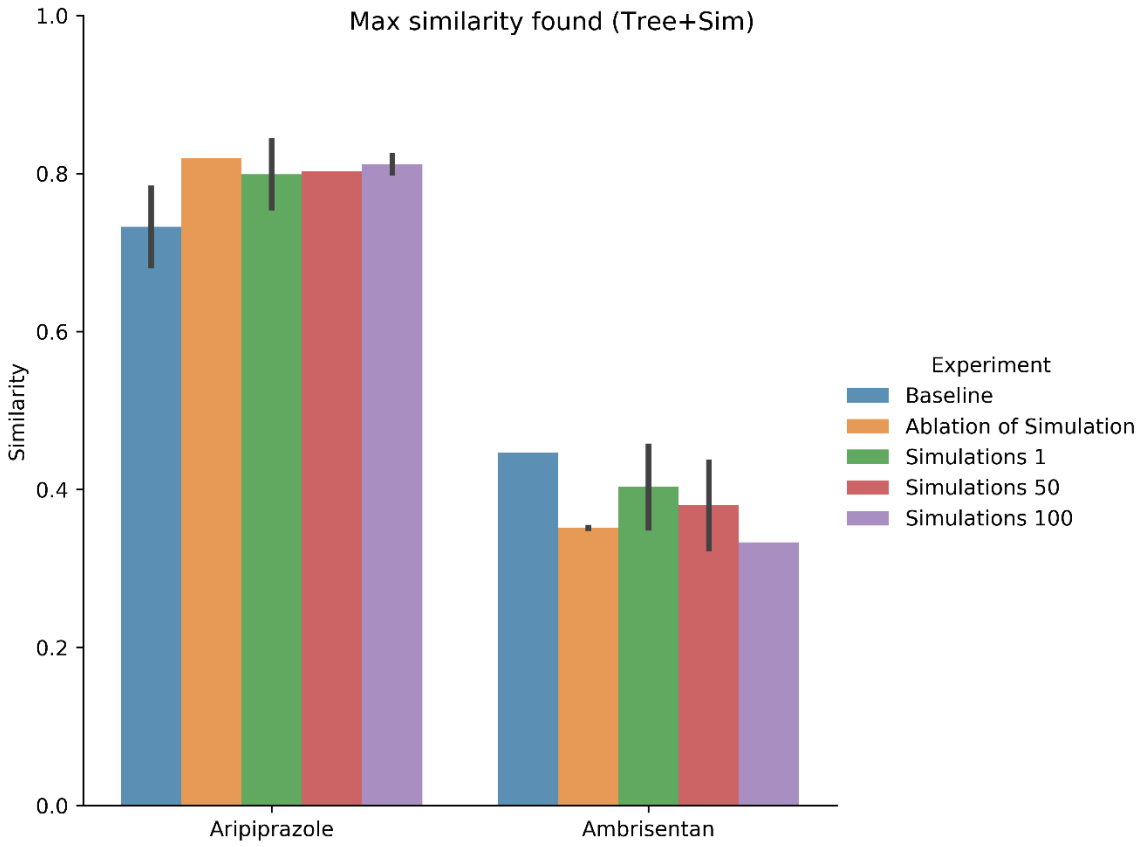


Figure 8-2 Maximum similarity found in the tree and simulation. The height of the bar is the mean, the black bars is ± 1 standard deviation.

8.4.3 Average score found in the tree

The average score found in the tree helps understand how the tree has moved through the search space. For example, high average scores suggest the search has focussed on regions in the search space with high structural similarity to the goal molecule. Figure 8-3 shows that the average score with random simulations is improved over the baseline for aripiprazole but not for ambrisentan. This would suggest that in some cases, the average quality of the tree can be improved by using an average returned from the simulation. However, it is not possible to draw universal conclusions from these two example problems.

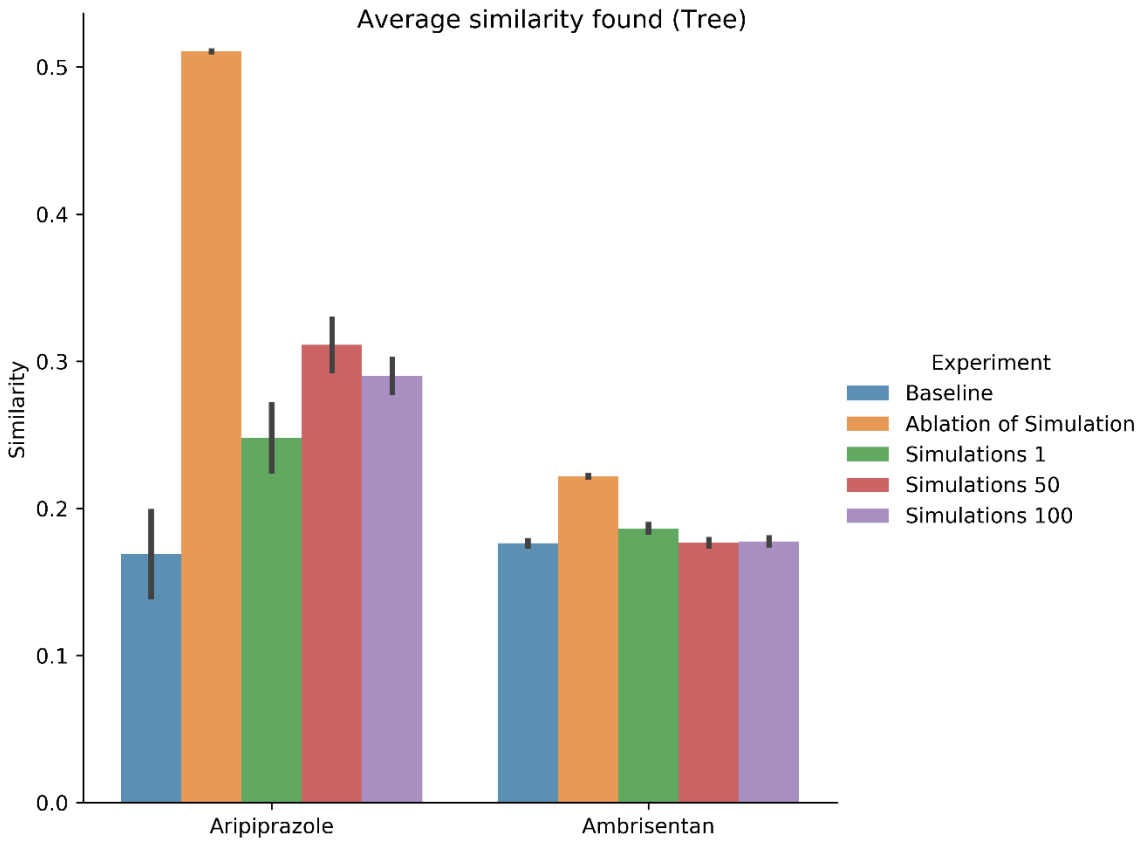


Figure 8-3 Average score of the tree. The height of the bar is the mean, the black bars is ± 1 standard deviation.

8.4.4 Skewness

Skewness is a measure of the direction of the tail of the distribution. If the skewness is negative, the tail points towards zero. If the skew is positive, the distribution points towards one. Figure 8-4 shows the skewness of the searches. The simulations returned a positive skewness in all cases, suggesting that the trees have not grown into regions of the search space with high scoring molecules.

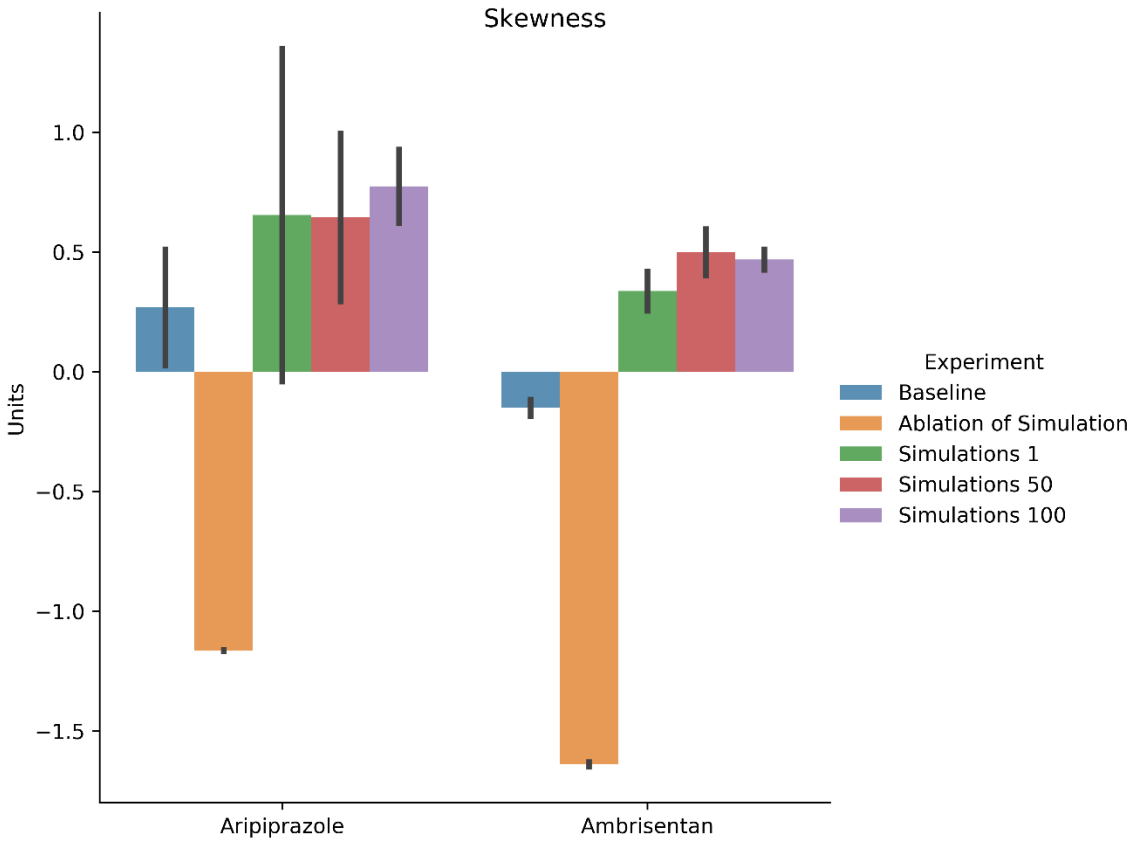


Figure 8-4 Skewness of the scores of the tree. The height of the bar is the mean, the black bars is ± 1 standard deviation.

8.4.5 Depth

Depth is a measure of the maximum number of reaction steps reached during the search. Deeper trees generally lead to higher scoring molecules due to the increased mass as more reactions are performed. Figure 8-5 shows the maximum depth reached by the tree. In the aripiprazole problem, the inclusion of the random simulation led to an increase in the overall depth to the specified maximum of three. A significant issue in ambrisentan, however, is that the maximum number of reaction steps prevents the tree from finding the goal. This issue is due to three reasons 1) There are too many molecules at each level of the tree, 2) the scoring is not discriminative enough, 3) there are not enough iterations. First, too many molecules mean the tree will naturally grow wide but not deep. Second, scoring is not allowing enough separation to force the selection to go down the same path over repeat iterations. Finally, with a wide tree and lack of separation in the scores, it could well be that the number of iterations simply limits the tree in its growth. It is important to note that in the case of the ablation of simulation, the depth that the tree can grow is not limited. In the simulation experiments, the depth is limited to (aripiprazole = 3 and ambrisentan = 4).

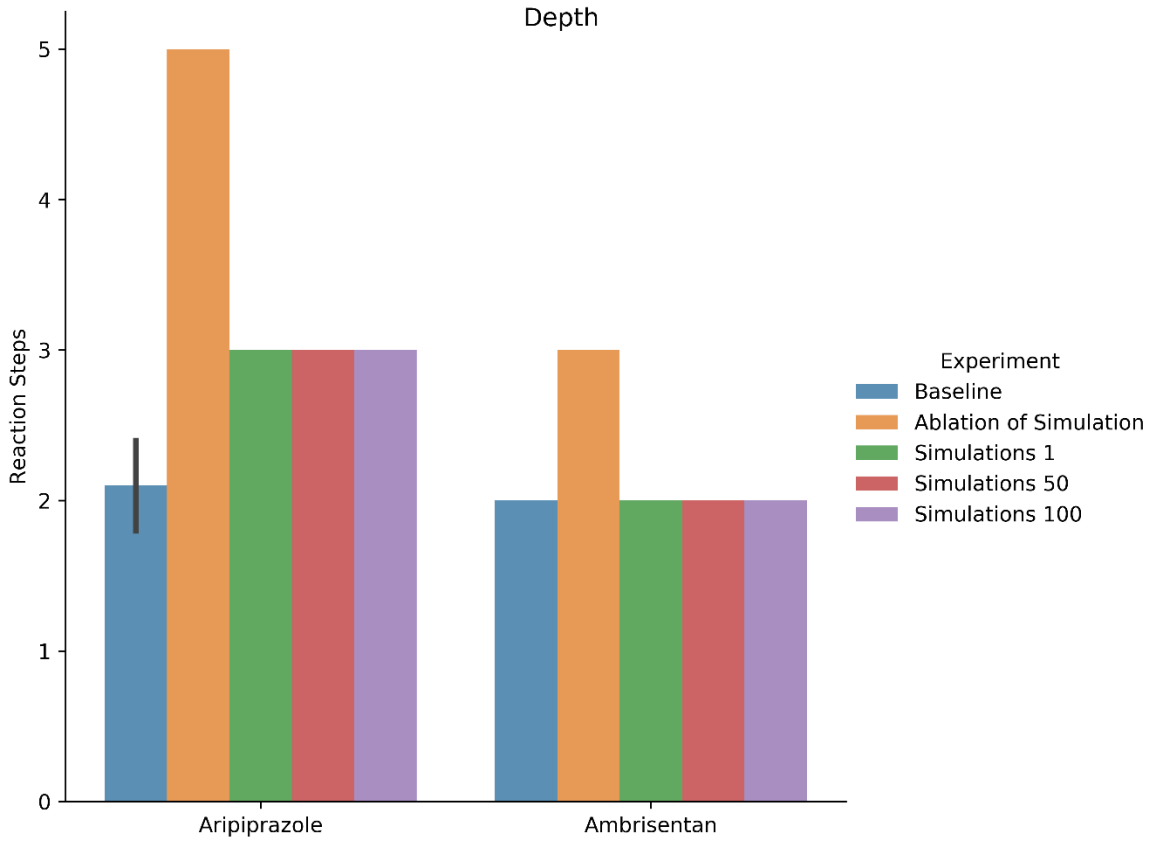


Figure 8-5 Depth of the search tree in reaction steps. The height of the bar is the mean, the black bars is ± 1 standard deviation.

8.4.6 Branching factor

The branching factor is a measure of the approximate number of child nodes per parent. High branching factors will lead to more unsatisfactory search performance due to the increase in the number of decisions to make at each level and the eventual computational expense due to more molecules needing to be simulated. Figure 8-6 shows the branching factors for each of the experiments. For the aripiprazole problem, there is an increase in the branching factor across the problems whereas in ambrisentan the trend of increasing branching factor is not observed. This increase in the branching factor is likely due to the restricted depth of the search. If the search reaches depth three in the aripiprazole case, then a zero value is backpropagated, and the visit count is increased by one. This encourages the search to explore more nodes at depth two, thus leading to a more significant branching factor.

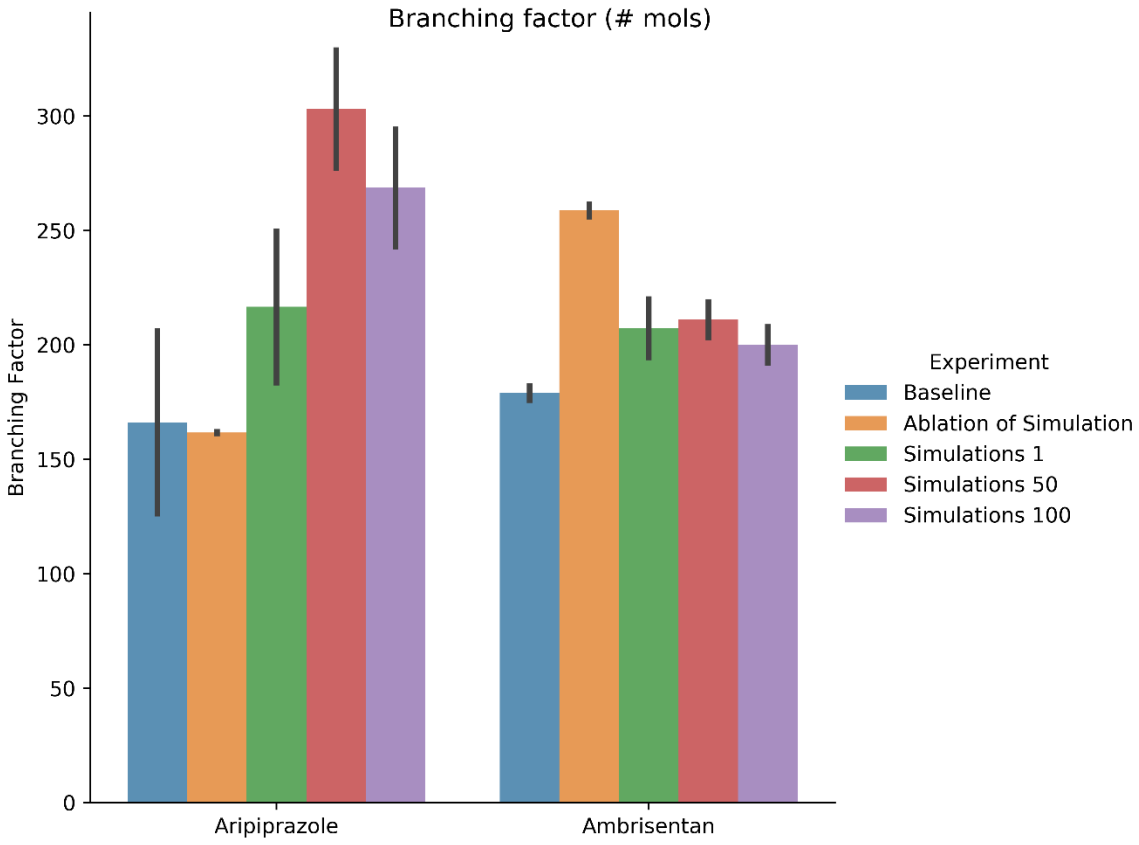


Figure 8-6 Branching factor of the tree. The height of the bar is the mean, the black bars is ± 1 standard deviation.

8.4.7 Time taken for the search to complete

Time taken is an essential metric for understanding search performance. Figure 8-7 shows that the total time with the simulations is dramatically increased from less than 60 minutes for the baseline for aripiprazole to just under 500 minutes for the simulation with 100 endpoints. The increase in time for the ambrisentan problem is even more significant, with the simulation with 100 endpoints case taking upwards of 2500 minutes (41.6 hours) to complete. The increase in the total time taken is due to the increase in database transactions and the total number of molecules generated. The substantially larger increase in time for ambrisentan is because of the increased depth required (four compared to three), which means that more molecules and more database transactions are needed. In addition, the computational cost of simulation for just 50 iterations prevents longer searches from running, effectively locking the tree to its shallow growth.

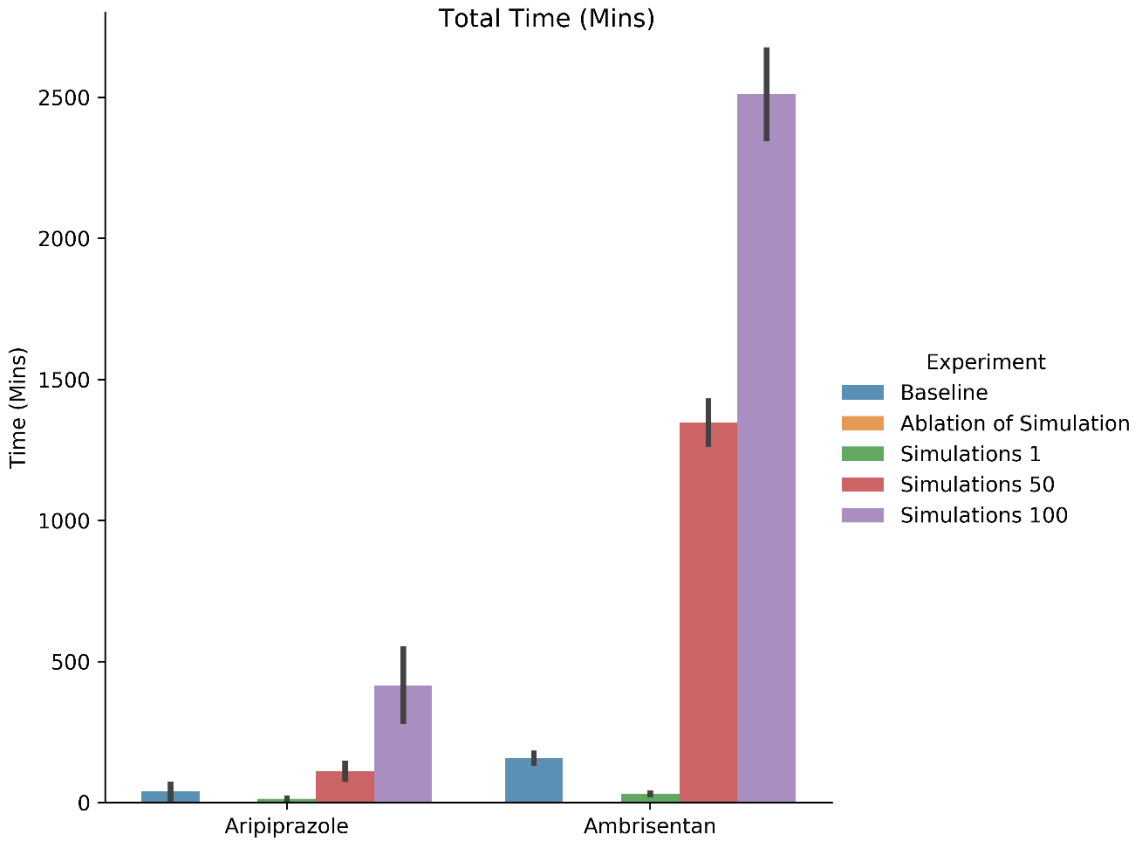


Figure 8-7 Total time. The height of the bar is the mean, the black bars is ± 1 standard deviation.

8.4.8 Tree size

Tree size is the total number of molecules generated during the search. Tree size will influence the amount of computational resources used and the total memory that will be allocated. Figure 8-8 shows the total tree size for the searches. The ambrisentan problem showed a slight increase in the overall tree size compared to the baseline. Interestingly the simulation with a single endpoint yielded a large tree comparable in size to the ablation of simulation. This suggests that the tree being grown is expanding by being directed away from dead-end unexpandable molecules.

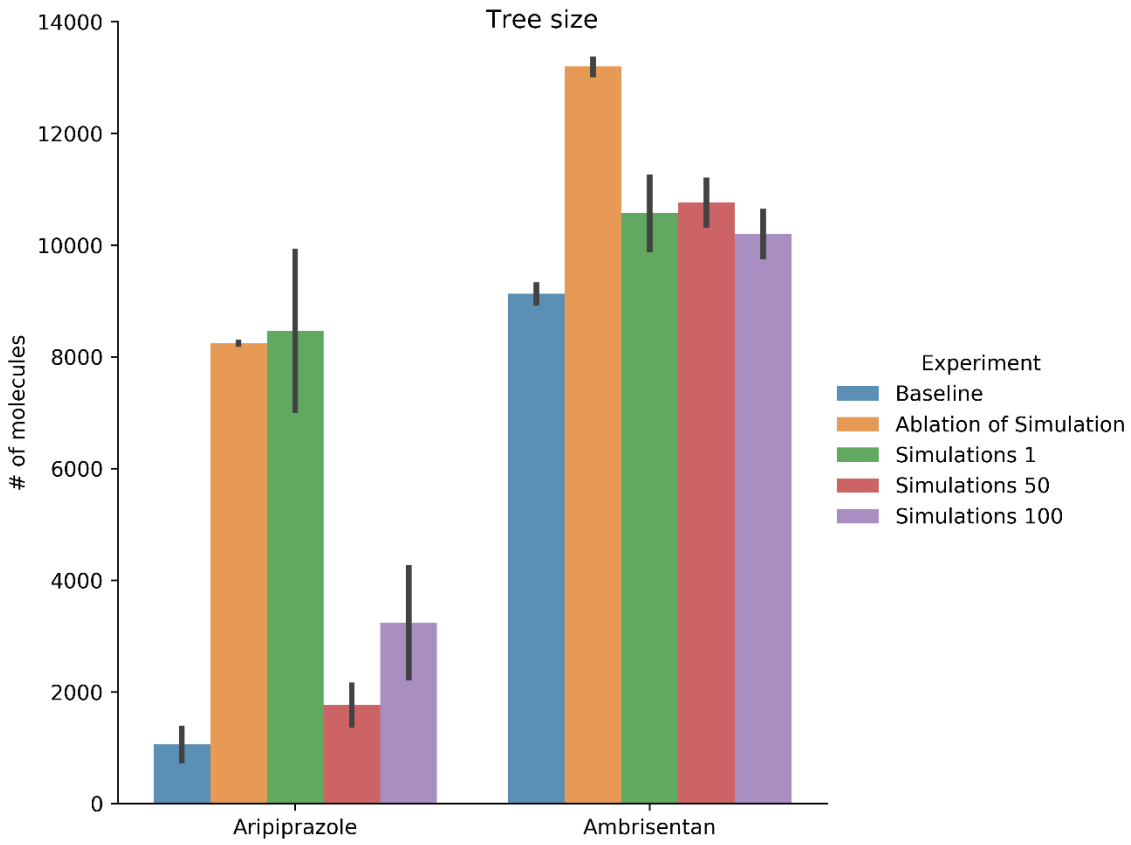


Figure 8-8 Tree size. The height of the bar is the mean, the black bars is ± 1 standard deviation.

8.4.9 Tree visualisation of the searches

Visualisation of the tree searches is helpful to interrogate the search performance qualitatively. The search tree is visualised using the same process as described in Chapter 7. Just one replicate is shown for each configuration and each problem.

8.4.9.1 Visualisation of the aripiprazole searches

The trees show slight variation overall. The main difference is that the total number of expanded molecules in the single endpoint tree is higher than that of the other two trees. This would suggest that overall, the single endpoint is causing more exploration to be carried out. However, a deep subtree present in all trees can be seen beneath the high scoring intermediate shown on the far right of level one. This would suggest that in all searches, there is some greedy selection of molecules with high scores; however, as this is only a single replicate on one problem example, it is hard to generalise from one replicate. Interestingly for the 50 and 100 endpoint trees, there is no discernible difference in overall tree shape and only a slight decrease in the number of molecules being expanded from level one. Therefore, it is likely, that the search is getting trapped in a dead-end unexpandable molecule.

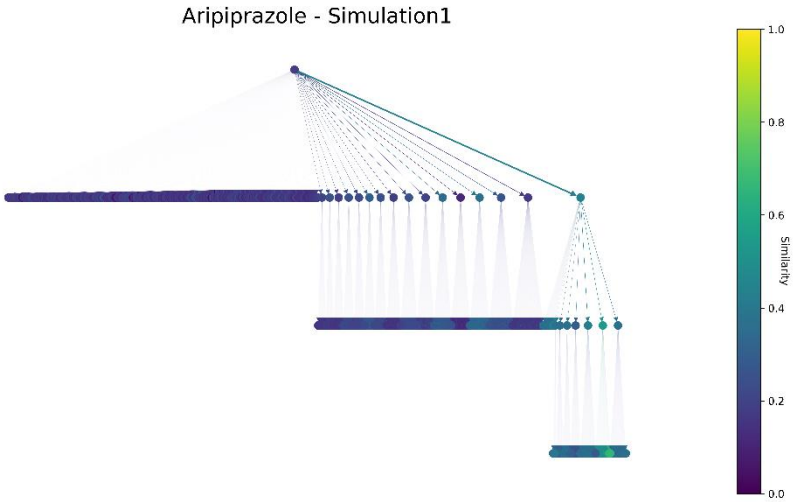


Figure 8-9 Tree visualisation of aripiprazole simulation-1 search

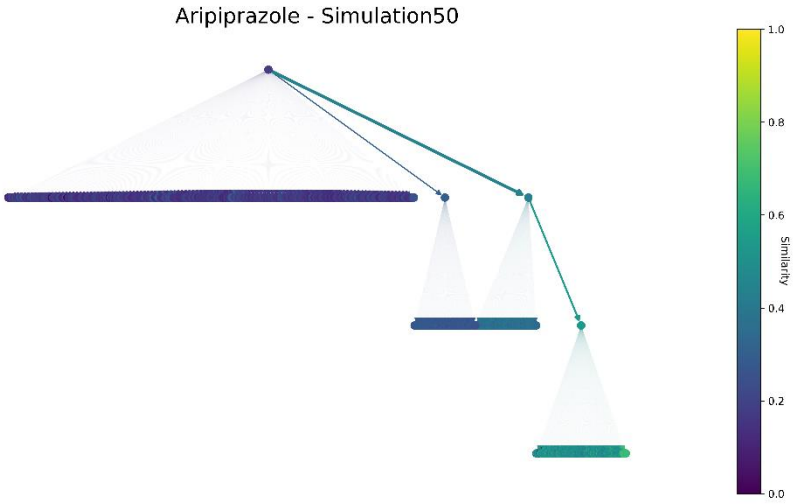


Figure 8-10 Tree visualisation of aripiprazole simulation-50 search

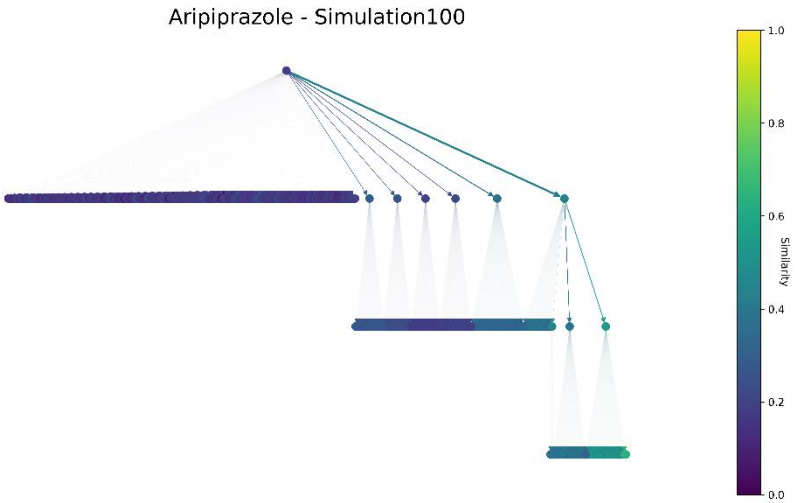


Figure 8-11 Tree visualisation of aripiprazole simulation-100 search

8.4.9.2 Visualisation of the ambrisentan searches

There are no discernible differences between the searches.

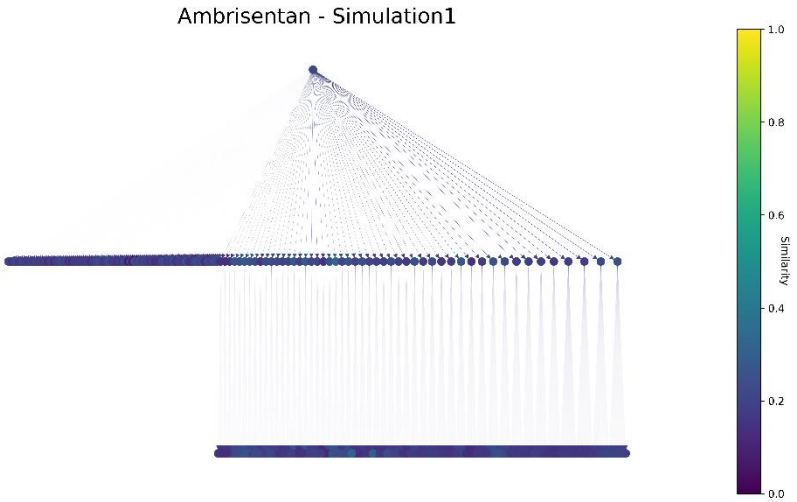


Figure 8-12 Tree visualisation of ambrisentan simulation-1 search

Figure 8-13 Tree visualisation of ambrisentan simulation-50 search

Figure 8-14 Tree visualisation of ambrisentan simulation-100 search

8.5 Conclusions

The random simulation experiments demonstrated that for the RVMCTS, a random simulation does not provide performance improvements over the baseline (depth one full enumeration). Although a performance benefit was seen in the molecules generated in the aripiprazole tree, there was no improvement when considering molecules generated during the simulation. Moreover, the performance benefit was not present for the ambrisentan problem. The tree visualisations indicate that the simulations lead to greedier searches, likely due to the additive nature of RVs maintaining high scoring intermediate substructures. A significant issue with simulation is the enormous increase in the total time taken. This is due to many database transactions and the number of attempts required to reach the desired number of endpoints (x100). Finally, due to this extreme increase in the amount of time required to run a random but deeper search, the number of iterations possible is effectively capped at 50 iterations reducing the amount of benefit from reinforcement learning.

In the next section, a machine learning approach to simulation is explored. The machine learning approach is explored to overcome the computational expense of the simulation step of the RVMCTS.

8.6 ML-based simulation

The problem of computational expense of simulation in the MCTS is not uncommon and in the alpha zero algorithms (Schrittwieser et al., 2020; Silver et al., 2018) a regression function was introduced as a replacement for simulation. Essentially, the simulation score of a node is replaced by a regression function from a complex deep neural network. As highlighted previously, a simulation is simply a heuristic built "on the fly" through learned experiences. For example, a simulation in the RVMCTS is designed to estimate the distance of a given molecule to a final goal molecule. From the experiments in the first part of this chapter, the computational expense of building the classical random look ahead was considerable. Therefore, either the random look ahead must be replaced with a computationally less demanding simulation approach, or the number of states to simulate at each stage needs to be reduced. In this section, the use of a machine learning-based heuristic is explored.

Building an ML-based heuristic function requires large amounts of data. Two approaches have become popular, offline(Maddison et al., 2015; Nasu, 2018) and online (Silver et al., 2018; Wang et al., 2020). When a large corpus of examples is available, offline learning can be used to learn a heuristic function. Online learning is where an experience buffer is created, and results are added to the buffer as the search is run, and a heuristic function is learnt from the buffer. This process is iterative, with the heuristic function being updated as more data is added to the buffer. Therefore, the main difference between offline and online learning is that the heuristic function in the offline approach is learnt once and is not updated. In online learning, the scoring function is learnt over successive cycles of the algorithm. The replacement of simulation by the machine learning approach is shown Figure 8-15 the updated schema of the tree search below.

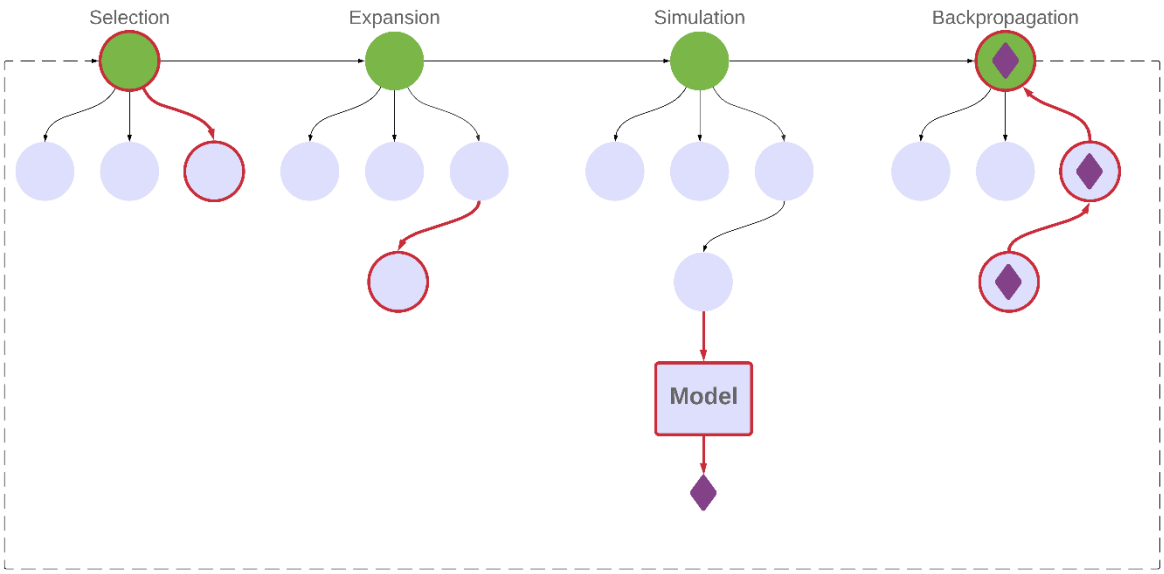


Figure 8-15 Here is an updated figure of the tree search. Rather than a random lookahead to set depth. Instead, the node is passed into the model, and a value is returned. This value is then backpropagated up the tree.

In this work, offline learning is explored and is described in the following section.

8.7 Methods

Building a machine learning model to replace the simulation component of the tree search requires training data. Training data comprises input features and corresponding output variables. The goal is to learn a function that returns a continuous value representing the expected output from a single depth one full enumeration. Depth one full enumeration was chosen for several reasons: 1) It allows for a comparison with previous approaches 2) The computational expense of traversing a deeper tree is computationally expensive. 3) if a large enough sample of molecules is expanded with various complexity (fragment, lead, drug), then a robust heuristic function can be learned. Therefore, an appropriate output variable needs to be selected. The output variable should be continuous and bounded between zero and one. Furthermore, when considering the random look ahead approach explored earlier in the chapter, the value returned from simulation to back-propagate was the mean subtree that existed beneath the node. Here, several different approaches to building a corresponding output variable are explored, such as Mean, Geometric Mean and Max.

8.7.1 Building the training data

The following approach is taken to build the training data:

1) Take a SM and apply all possible transformations.
2) The generated molecules are then scored.
3) A group by per SM molecule is performed computing a given value per SM.

A visual representation of the above training process is given in Figure 8-16:

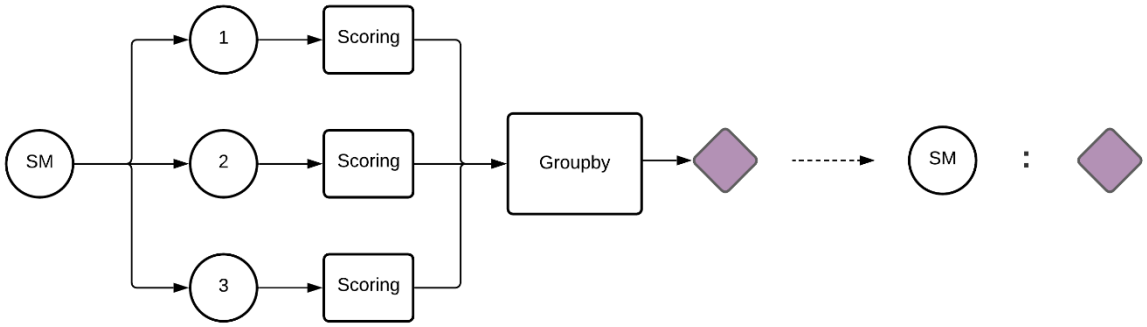


Figure 8-16 The SM is expanded. Each product generated is scored. The scored values are then merged using a group by function. The training data is formed such that the merged score is the out variable and the SM is the input features.

8.7.2 Training data generation

300,000 molecules were drawn from ZINC20-in-stock and expanded to depth one using the JMedChem 4.5 K seeded database. The SMs were 100,000 fragments, 100,000 leads, 100,000 drugs sampled randomly from ZINC20 in-stock. The expansion was carried out on the Sheffield high-performance computing cluster (SHARC), the computation was spread out over 200 compute nodes at once. In total, this generated 32.43M compounds; 427k invalid molecules were removed due to not being read by RDKit (SM or structure). In addition, 4.25M duplicate SM and product pairs were removed. This left 27.7M readable products (shown in Figure 8-17).

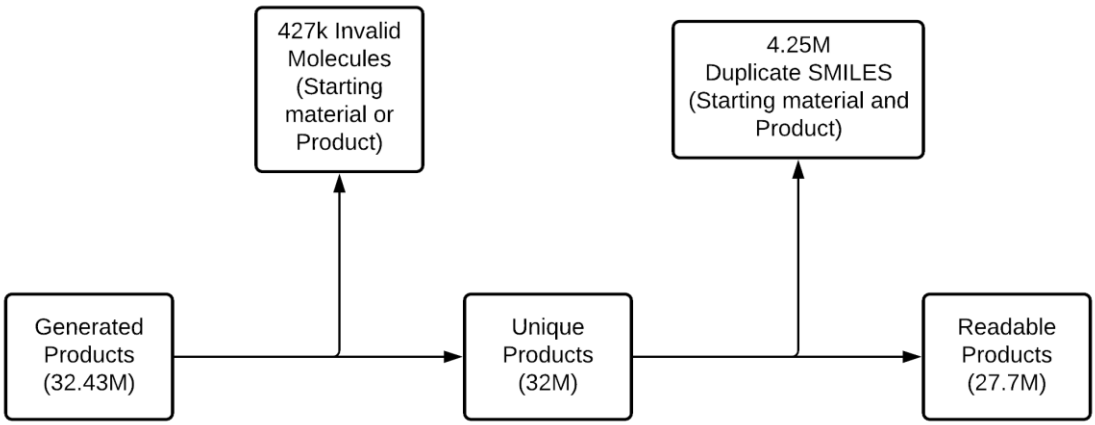


Figure 8-17 Workflow to clean generated molecules, 427k invalid molecules removed and 4.25M duplicate SMILES SM and product pairs, returning 27.7M products

The 27.7M compounds were then scored using once using the corresponding similarity objective score. Following scoring, the data is then grouped by SM and summarised per product scores into three separate groups (Mean, Geometric Mean and Max). In total, this pre-processing led to 253k SMs from the starting dataset of 300k compounds. This process is repeated for both aripiprazole and ambrisentan objective functions.

8.7.3 Model training

Linear regression was used to build the models using the Vowpal Wabbit (VW) machine learning package (*Vowpal Wabbit*, 2021). The linear regression uses stochastic gradient descent to fit the model. VW is attractive for this problem for several reasons: sparse representation and online learning. Sparse representation allows the dataset to be converted into a representation that stores only the non-zero features. This is done by keeping track of non-zero values in the input data at each training step so that, in the case of large sparse inputs, most of the values are ignored as they are zero. Online learning in this context allows the data to be read and used as training examples without the need to store the entire dataset in memory . Online learning is vital as it allows for larger feature set sizes to be used. In contrast more complex models such as decision trees are not trainable on the large amount of data required to effectively learn the mapping of state to simulation outcome as they require operations to be carried out on the total dataset. Finally, while it is obvious that there are shortcomings associated with linear models, due to the size of the dataset and the requirement of training on large dimensional sparse fingerprints, the shortcomings are accepted and there are few alternatives without the overhead of more computation.

The default parameters for the VW linear regression are used and shown in Table 8-3 below:

Parameter	Value
Training method	Stochastic gradient descent with adagrad
Initial learning rate	0.5
Loss function	Squared

Table 8-3 Linear regression model training parameters

The next step of the model building process is to convert the SM molecules into a machine learnable representation. Morgan RDKit fingerprints were used with lengths 1024, 2048, 4096, 8192 and 16384 bits, and with radii two and three. Morgan fingerprints were used as they have demonstrated competitive performance in a wide range of machine learning tasks (Robinson et al., 2017). To identify the best performing fingerprint length and radius, the dataset was split at random into a training set and a testing set with ratio 90:10, respectively. For the training data, 10-fold cross-validation was performed for each fingerprint. The cross-validation results for aripiprazole and ambrisentan models are shown in Table 8-4 and Table 8-5. The values correspond to the mean R^2 score of the cross-validation along with a corresponding standard deviation with the best values shown in bold. Each fold, took two minutes to train utilising a standard consumer laptop (Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.59 GHz, 32.0 GB). Each R^2 took approximately 20 minutes to compute.

	Aripiprazole Mean	Aripiprazole GMean	Aripiprazole Max
1024 bits 2 radius	0.730 ± 0.004	0.640 ± 0.027	0.640 ± 0.003
1024 bits 3 radius	0.675 ± 0.004	0.572 ± 0.016	0.570 ± 0.005
2048 bits 2 radius	0.769 ± 0.005	0.687 ± 0.007	0.715 ± 0.002
2048 bits 3 radius	0.713 ± 0.003	0.622 ± 0.009	0.631 ± 0.002
4096 bits 2 radius	0.776 ± 0.003	0.739 ± 0.005	0.762 ± 0.002
4096 bits 3 radius	0.737 ± 0.003	0.672 ± 0.007	0.693 ± 0.005
8192 bits 2 radius	0.787 ± 0.004	0.771 ± 0.004	0.777 ± 0.002
8192 bits 3 radius	0.746 ± 0.004	0.696 ± 0.005	0.707 ± 0.006
16384 bits 2 radius	**0.795 ± 0.003**	**0.796 ± 0.004**	**0.798 ± 0.003**
16384 bits 3 radius	0.764 ± 0.004	0.743 ± 0.006	0.757 ± 0.003

Table 8-4 Mean R^2 values for the 10-fold cross-validation with a corresponding standard deviation of the mean (Aripiprazole).

	Ambrisentan Mean	Ambrisentan GMean	Ambrisentan Max
1024 bits 2 radius	0.643 +- 0.012	0.581 +- 0.028	0.493 +- 0.009
1024 bits 3 radius	0.597 +- 0.009	0.538 +- 0.022	0.400 +- 0.009
2048 bits 2 radius	0.673 +- 0.007	0.600 +- 0.017	0.572 +- 0.011
2048 bits 3 radius	0.630 +- 0.007	0.561 +- 0.011	0.454 +- 0.013
4096 bits 2 radius	0.681 +- 0.011	0.617 +- 0.013	0.617 +- 0.014
4096 bits 3 radius	0.623 +- 0.011	0.536 +- 0.008	0.532 +- 0.012
8192 bits 2 radius	0.689 +- 0.009	0.628 +- 0.011	0.641 +- 0.008
8192 bits 3 radius	0.623 +- 0.007	0.549 +- 0.005	0.568 +- 0.009
16384 bits 2 radius	**0.700 +- 0.007**	**0.645 +- 0.008**	**0.689 +- 0.007**
16384 bits 3 radius	0.658 +- 0.006	0.600 +- 0.005	0.620 +- 0.009

Table 8-5 Mean R^2 values for the 10-fold cross-validation with a corresponding standard deviation of the mean (ambrisentan).

After identifying the best R^2 score (length=16384, radius 2), the models were then retrained using the entire training data and then tested on the holdout test set. The performance on the holdout set is shown in the Table 8-6 below:

	Mean	GMean	Max
Aripiprazole (16384 Bits,r=2)	0.794	0.794	0.801
Ambrisentan (16384 Bits,r=2)	0.702	0.651	0.680

Table 8-6 Final trained model R^2 on test data

Overall, the performance of each validated model is promising. One important consideration when replacing the simulation is the overall time required to run the search replacement. Using the holdout test set, an inference time for using the model is measured and is shown in Table 8-7 below.

	Max(ms)	Mean(ms)	GMean(ms)
Aripiprazole (16384 Bits,r=2)	1.3 +- 0.5	1.4+-0.5	1.4+-1.4
Ambrisentan (16384 Bits,r=2)	1.3+-0.5	1.4+-0.5	1.4+-0.7

Table 8-7 Mean inference times with a corresponding standard deviation of error.

Considering the validated R^2 and the corresponding time performance, this replacement for the simulation can now be integrated into the search.

8.7.4 Implementation of ML Simulation

The ML model is called in the "simulation" step as shown in Figure 8-15 and a value returned at simulation time, rather than generating molecules via either the full enumeration described in the previous chapter or the random simulation described earlier in this chapter. This value is then backpropagated.

8.7.5 Experiments

Three different ML models (Mean, GMean, and max) were investigated for both aripiprazole and ambrisentan with five repeats for each run.

8.7.5.1 Search parameters

The same search parameters were used as the lookahead simulation experiments earlier in this chapter, including a fixed depth (aripiprazole = 3, and ambrisentan = 4).

8.8 Results

The results of the searches are compared to the baseline performance of the depth one full enumeration and the ablation of simulation described in the previous chapter.

8.8.1 Maximum score

The maximum score found during the search reveals the overall performance of the tree during the search. Figure 8-18 below shows the maximum score found in the tree. Across the ambrisentan problems, there was no significant improvement in maximum score over the baseline or the ablation of the simulation experiment. In the aripiprazole problem, there was an improvement in the maximum score seen over the baseline and comparable performance to the ablation of simulation experiment. The highest score amongst the ML methods, however, was produced by the ML-Mean simulation. ML-Mean is where the mean of the child scores are aggregated, and the model is trained to predict the mean score of the parent's children. The high ML-Mean score is the same as the ablation of simulation. In practice, the mean of the subtree beneath a given node for the ablation of simulation (objective function Tanimoto similarity) will be equivalent to the ML-mean predicted score. Therefore, it is unsurprising that the same maximum scores are observed.

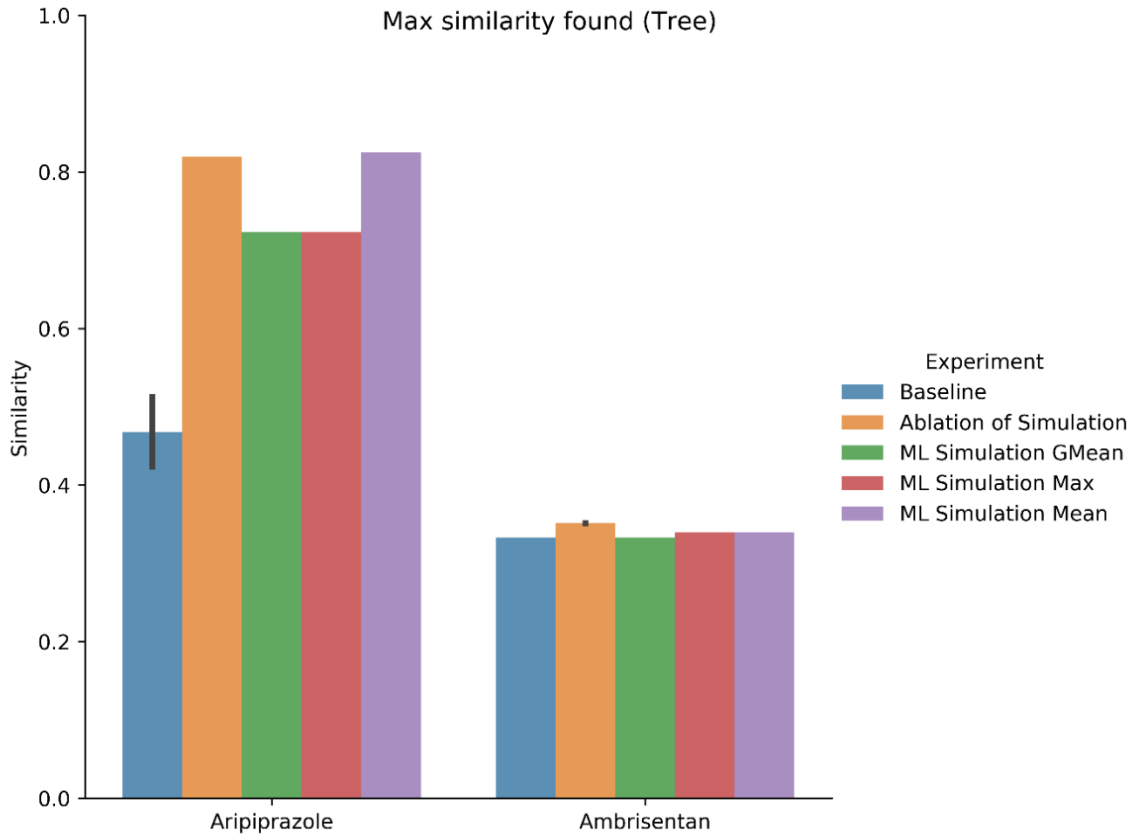


Figure 8-18 Maximum Similarity found during the search. The black bars represent ± 1 standard deviation present in the data. No black bar represents a standard deviation of zero.

8.8.2 Average score found

The average score found during the search helps understand the overall quality of the molecules generated during the search. Figure 8-19 shows the average similarities of the trees. In the ambrisentan example, all ML methods outperformed the baseline. In contrast, for the aripiprazole problem, the average score of all ML-based simulations is lower than the ablation of simulation. In the ML Max model, the performance was substantially lower than the ablation of simulation and was equivalent to the baseline approach. Suggests that the ML-max model moves into regions of the chemical space, which, on average, do not score highly but could give higher maximum scores. The same trend is also seen to some extent in the ambrisentan problem; however, the observed decrease is not substantial.

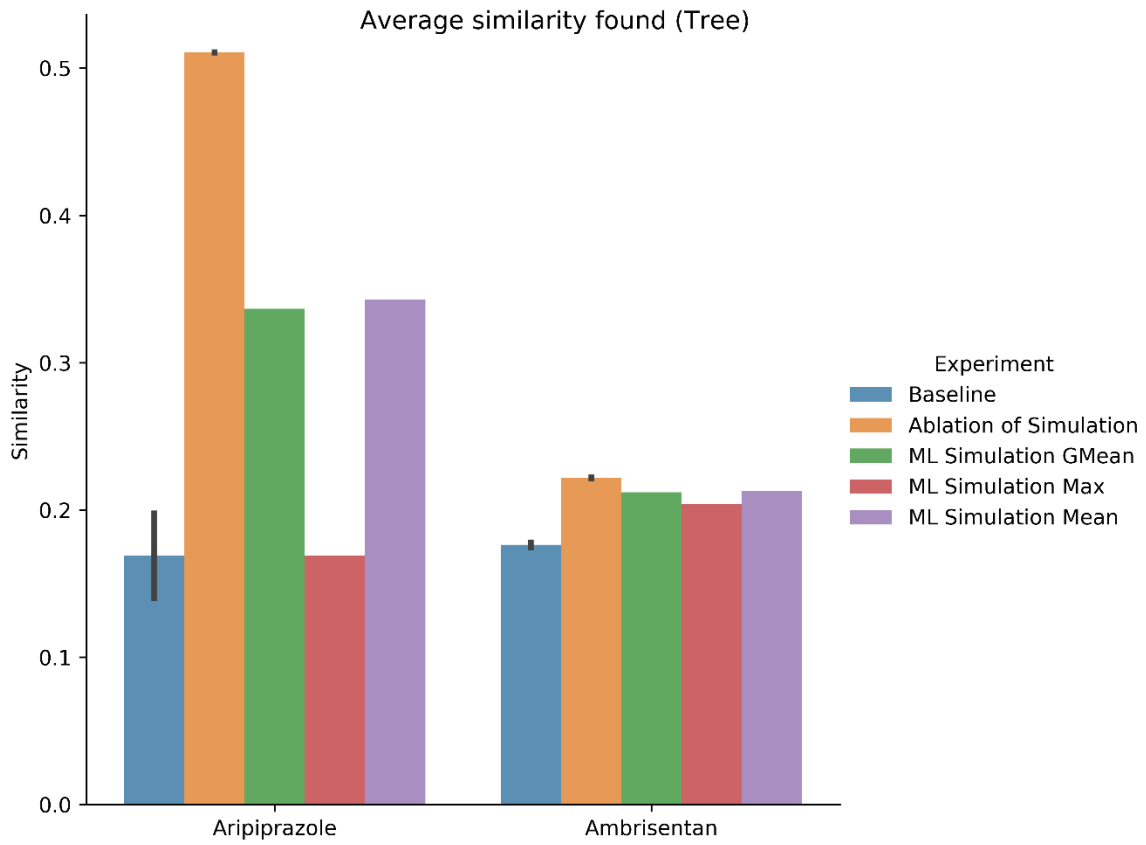


Figure 8-19 Average score of the tree. The black bars represent ± 1 standard deviation present in the data. No black bar represents a standard deviation of zero.

8.8.3 Skewness of the scores in the tree

The skewness of the scores in the tree demonstrates whether the tree has moved into better regions of chemical space. A negative value will suggest the distribution has a tail point towards zero, which is desirable for de novo design as the negative skew implies higher-performing compounds have been generated. Figure 8-20 shows the skewness of the distributions of the scores of the searches. In the aripiprazole problem, both the geometric and mean models led to a more negative skew of the distribution of the scores. In contrast, the ML-max model had a more positive skew compared to the baseline. The ML-Max model having a more positive skew is not surprising as the ML-model is trained to return the maximum score of the compounds, not improve the average. In the ambrisentan problem, the GMean and Mean ML models outperformed the ablated simulation slightly. This would suggest that more compounds are being generated with higher scores. However, when comparing with Figure 8-19, the increase in skew has not yielded a higher mean score for the ambrisentan ML models, just a change in the shape of the distribution.

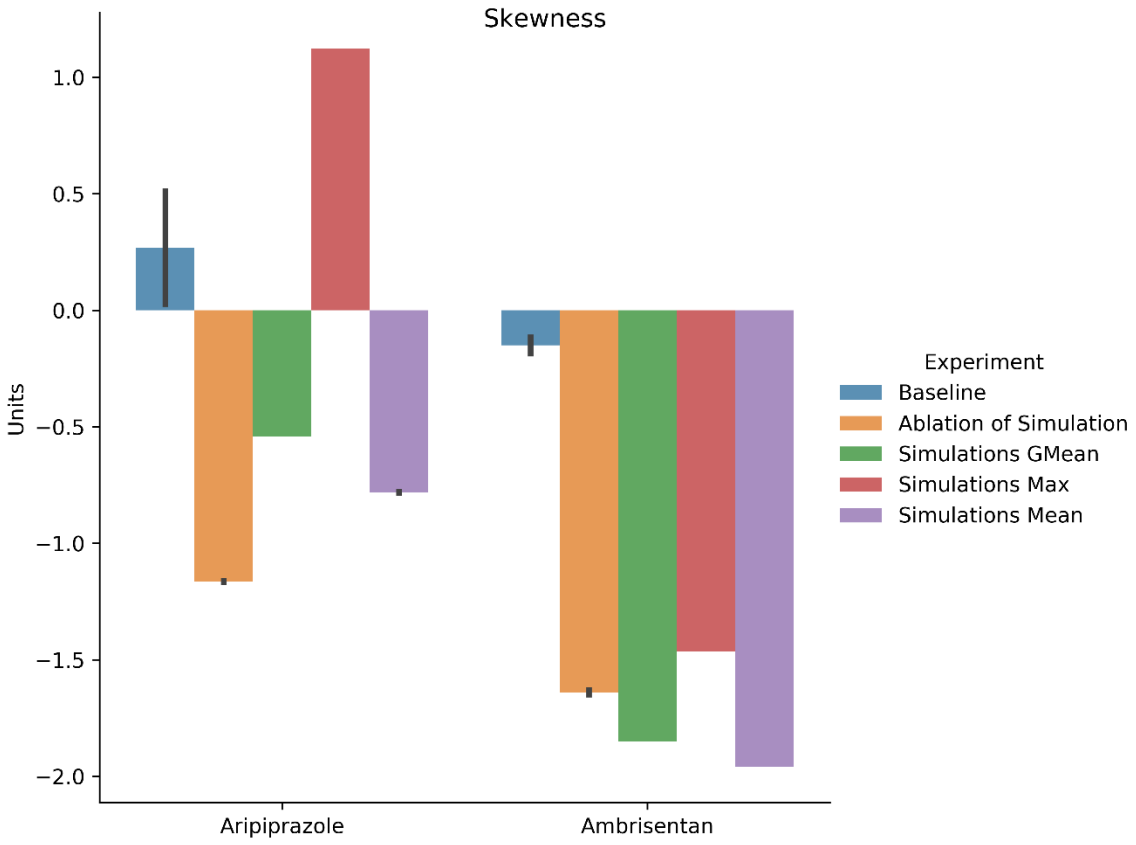


Figure 8-20 Skewness of the scores found in the tree. The black bars represent ± 1 standard deviation present in the data. No black bar represents a standard deviation of zero

8.8.4 Total time taken for the search to complete

The total time taken for the search to complete is shown in Figure 8-21 below. As expected, the ML model performs substantially better than the corresponding baseline. However, there is some variation across experiments, with the ML-Max model taking the longest amount of time. An increase in time taken can only be due to the increase in the number of molecules generated. Therefore, the results suggest that the ML-max model has generated a different tree with more leaves than the mean and GMean models.

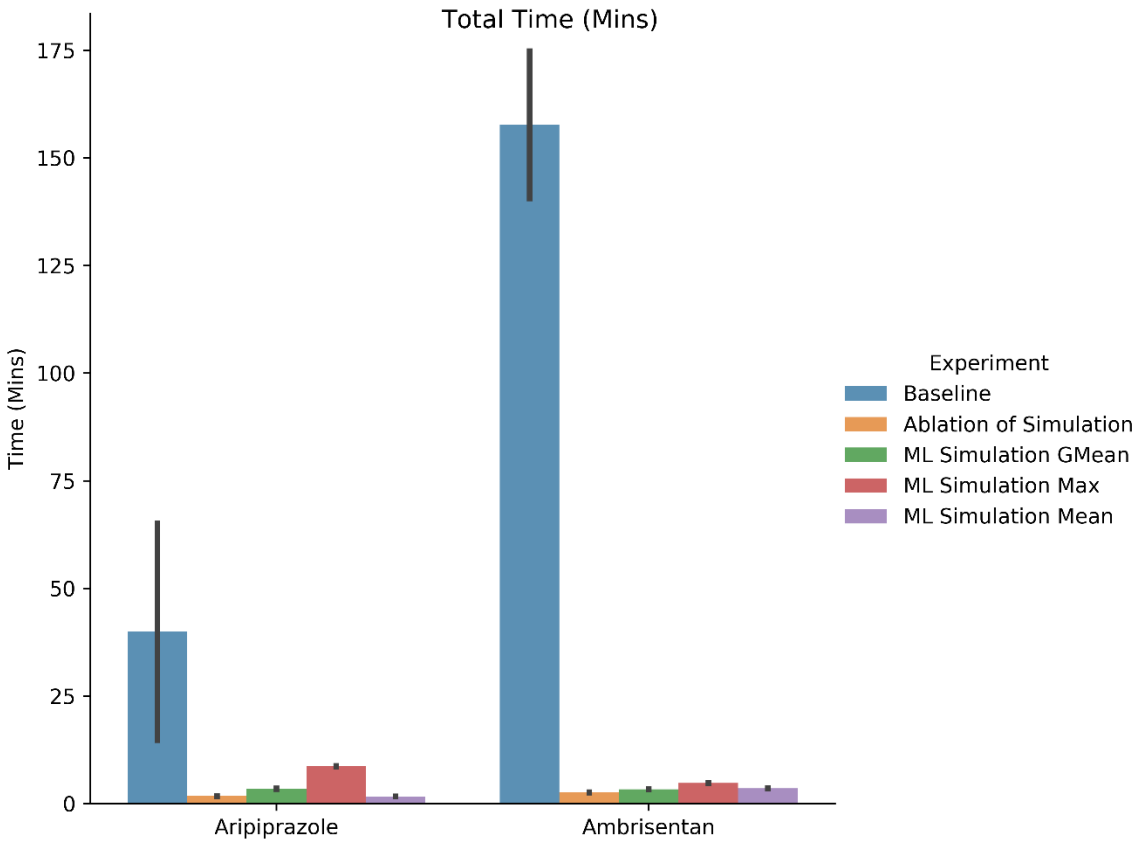


Figure 8-21 Total time taken for the search to complete in minutes. The black bars represent ± 1 standard deviation present in the data. No black bar represents a standard deviation of zero.

8.8.5 Branching factor

The branching factor is a measure of how many molecules are generated per SM molecule. In the aripiprazole example, the ML-max model had the highest branching factor. Compared to the other two experiments, the high branching factor would suggest that the ML-max model has yielded a search that has explored different regions of the chemical space compared to the GMean and Mean models. This higher branching factor is possible if the highest-scoring compound is different for Max compared to Mean and GMean. Interestingly, this is also true for the ML-GMean example, which also generated a different branching factor. In the case of the ML-mean model, the branching factor was low. The ML-max model still had a high branching factor in the ambrisentan problem compared to the other search configurations. However, in the ambrisentan example, ML-mean had a slightly higher branching factor than the ML-GMean search. Again, this would suggest that different trees are being grown and that changing the heuristic scoring function will change the type of tree being grown.

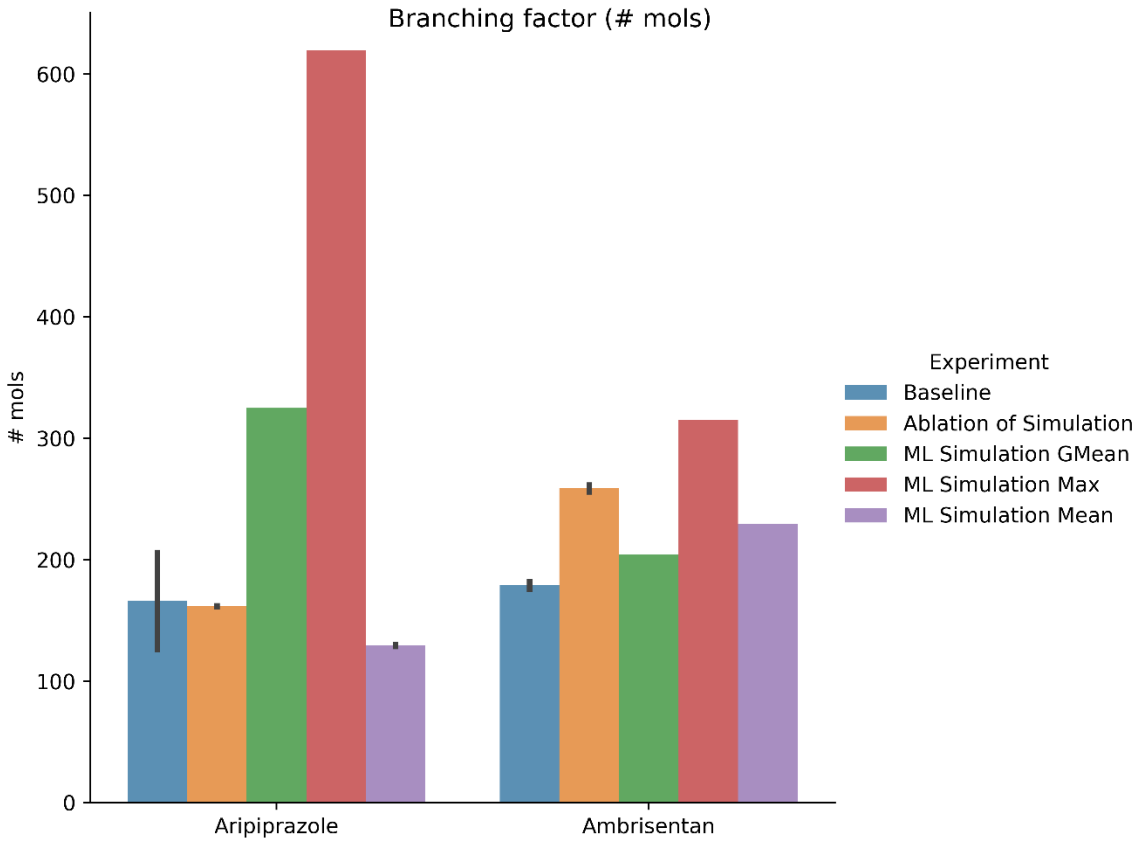


Figure 8-22 Branching factor for the search. The black bars represent ± 1 standard deviation present in the data. No black bar represents a standard deviation of zero.

8.8.6 Depth

Depth is a measure of how many reaction steps are being generated during the search. Figure 8-23 below shows the corresponding depth which each of the searches reached. The maximum search depth in the ML-model search experiments is restricted for both the aripiprazole (depth=3) and ambrisentan (depth =4). The search depth in the aripiprazole problems all reached the required maximum depth. However, in the ambrisentan problem example, only the ML-Mean model reached the required depth, the first time this has occurred so far in this work suggesting that the ML model is providing benefit to the search. The lack of depth means that the other searches could not have solved the ambrisentan problem. The lack of depth has been a consistent theme throughout all searches carried out in this work and will be addressed more thoroughly in Chapter 9.

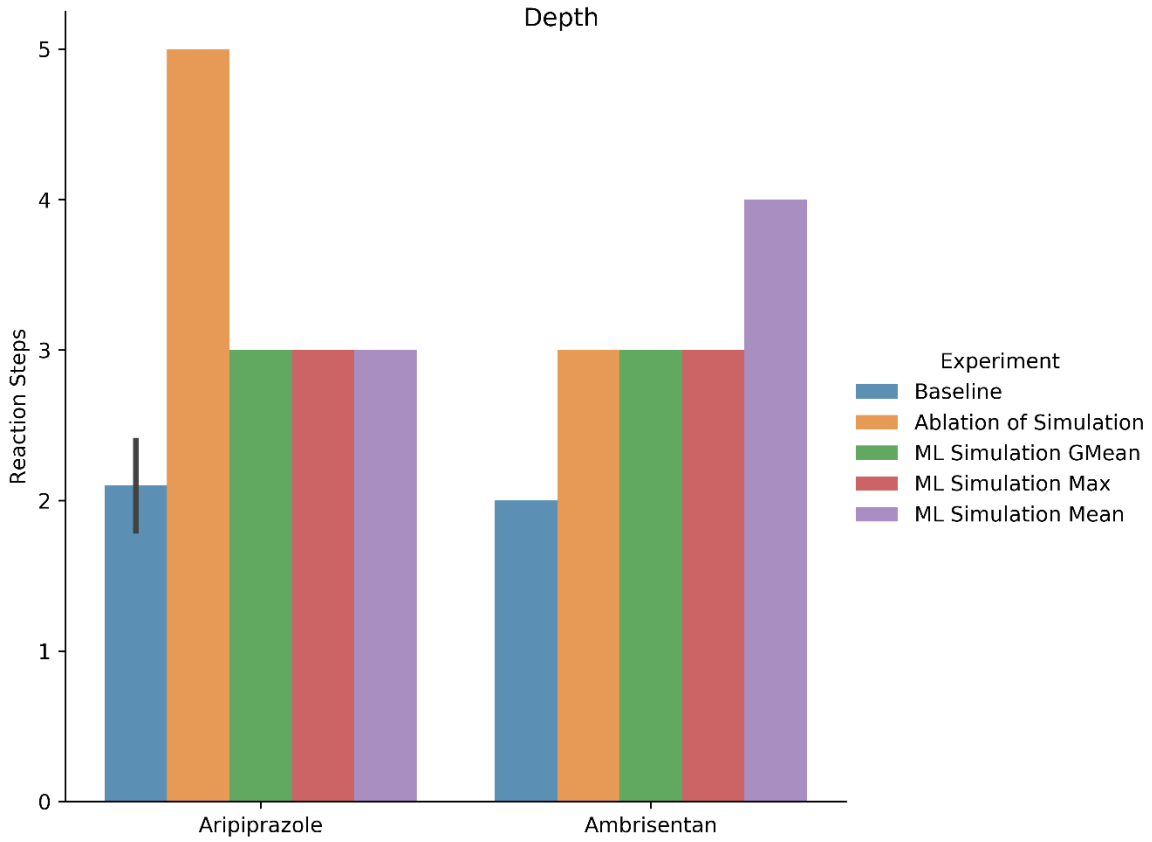


Figure 8-23 Depth of the search. The black bars represent ± 1 standard deviation present in the data. No black bar represents a standard deviation of zero.

8.8.7 Example trees

Visualisation of the tree searches allows for qualitative interrogation of the tree search performance. Again, for this work, one replicate was randomly sampled for each of the six experimental configurations. First, in the aripiprazole example, the trees show different amounts of exploration. In the ML-Mean, only one compound was expanded from depth one, and the rest were expanded at depth two. This suggests that the compound identified at level one was a strong performing intermediate which led the search to prefer that path greedily. In the ML-GMean problem, there was slightly more exploration, with more compounds being expanded on level one of the tree. However, in the ML-Max problem, a substantial amount of exploration occurred at level one of the tree with several nodes being expanded.

Interestingly in the ML-Max example, the score of the most selected node at level one (thickest line) was not high scoring in terms of similarity to the goal. The promotion of this node is promising as it suggests that the ML-max model predictions are not as strongly correlated with the SMs similarity compared to the other models.

In the ambrisentan problem, all of the models led to several molecules being expanded at level one with ML-Mean model showing the most significant amount of exploitation as the search descended into depth four.

8.8.7.1 Aripiprazole ML Trees

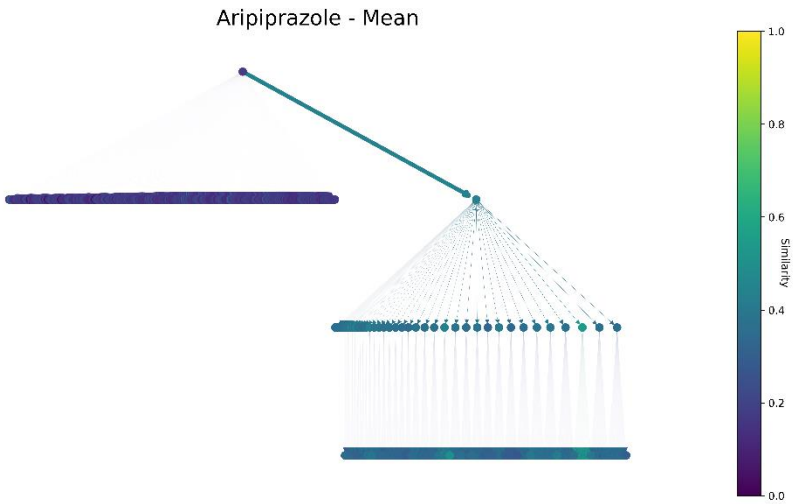


Figure 8-24 Aripiprazole ML-Mean tree

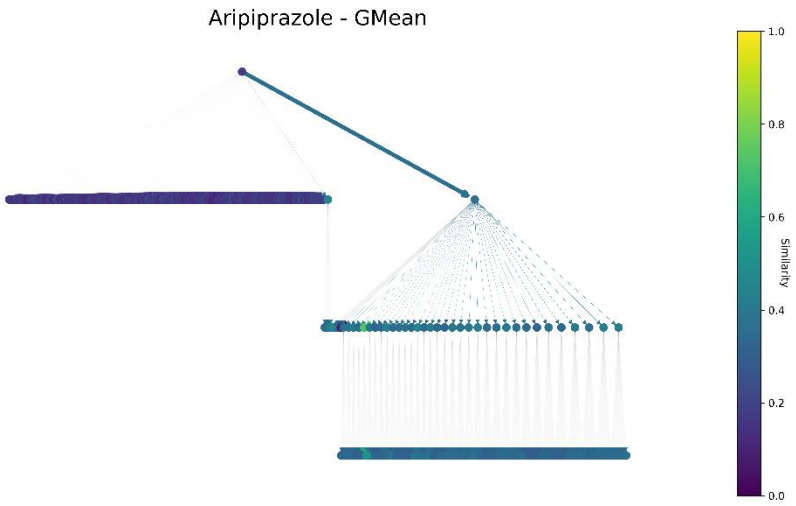


Figure 8-25 Aripiprazole ML-GMean Tree

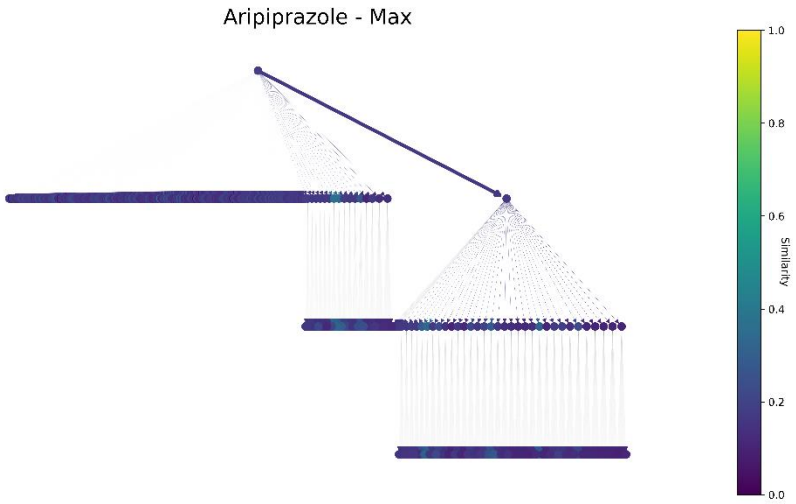


Figure 8-26 Aripiprazole ML-Max Tree

8.8.7.2 Ambrisentan ML Trees

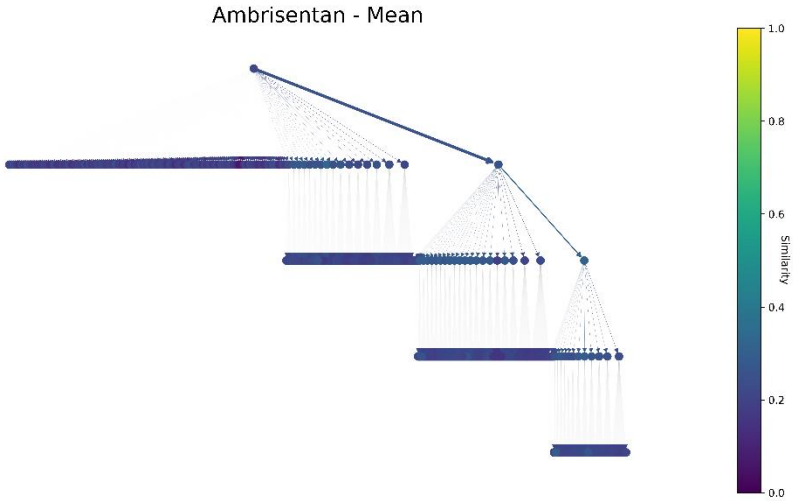


Figure 8-27 Ambrisentan ML-Mean Tree

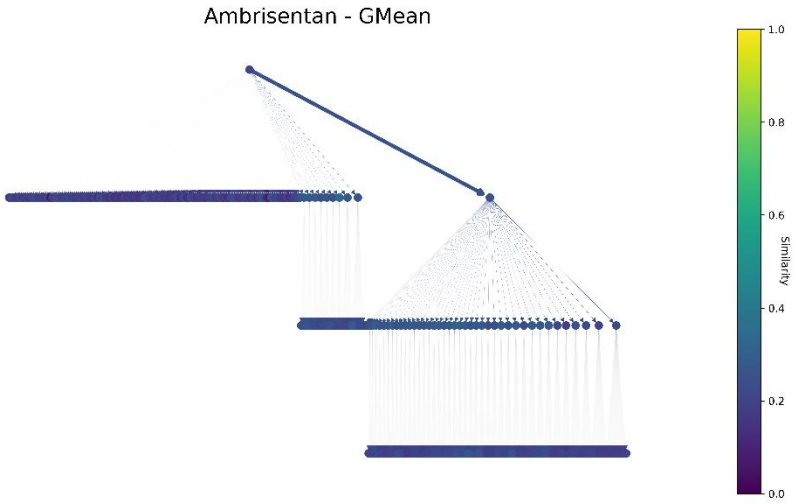


Figure 8-28 Ambrisentan ML-Gmean tree

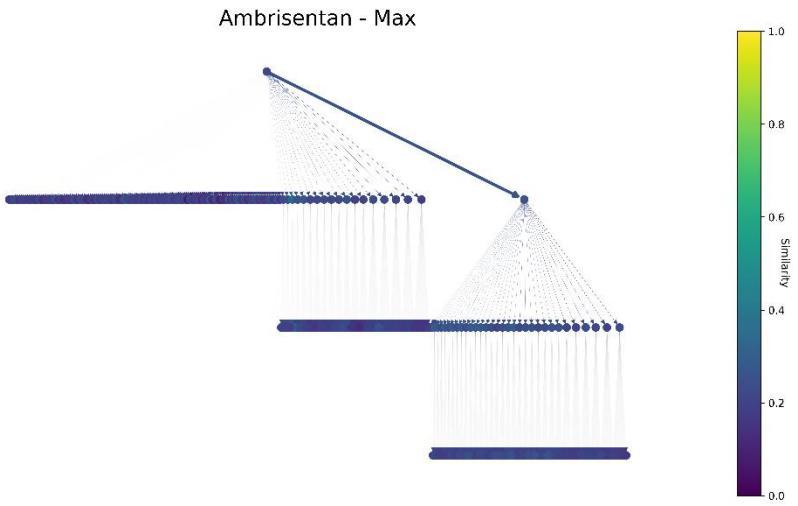


Figure 8-29 Ambrisentan ML-Max Tree

8.9 Conclusions

In the first part of this chapter, a series of experiments investigated the performance of a random simulation as part of the simulation component. The searches showed that the random simulation required a large computational expense with little overall benefit. In the second part of this chapter, a machine learning-based simulation was developed and tested across two benchmark problems (aripiprazole and ambrisentan). Overall, the performance of the ML-based approaches produced faster searches for all cases which were comparable to the performance of the ablated simulation approach explored during the ablation studies. Importantly the results here demonstrate that it is possible to learn a heuristic scoring function from generated data. Furthermore, the approach described here can be further extended to any scoring function (for example, docking) while maintaining an adequate constant time per simulation. This is crucial during applications where a computationally expensive scoring function might be used. Finally, significant performance improvements are not seen for model-based methods due to the limited number of iterations and the large number of generated molecules. A limitation, of the simulation model approach is that the selection of the models was based solely on the R^2 metric which can be limited in some circumstances. MAE and RMSE were not calculated as it is expected that the de novo design algorithm will ultimately traverse to regions of the search space out with the training data. This was an oversight, however, the only practicable approach to assessing the simulation model performance would be to select the model based on its combined ability with the tree search methodology. This would be possible by first shuffling the training data, training the model, and then using it within the search itself. After completing the search, the best performing molecule, would

then be used to assess whether the model has performed well. This process would be repeated N times on a set of training data to assess the model's overall performance. Finally, the model which yielded the highest performing molecule when combined with the search would be selected. This approach would be computationally demanding to perform as for each shuffle of the data, a model needs to be trained, and then a search run. Hence, the R^2 metric was employed here and, although it is limited, it was shown to provide enough assessment of model performance to drive the search to regions of interest.

The next chapter will explore further modifications of the tree search approach to minimise the number of compounds generated.

Chapter 9 Modifications

9.1 Introduction

The performance of the RVMCTS was explored in both Chapter 7 and Chapter 8. The results suggest that the RVMCTS can rediscover seeded routes, or generate molecules close to the target molecule, for some target molecules. However, the RVMCTS had some limitations which can be summarised in the following two key points: the number of molecules generated was too high; and the selection component of the RVMCTS was not tuned to force depth in the tree and the generation of more complex molecules. This chapter consists of three parts. First, an introduction to a series of MCTS modifications is given along with their implementations. Second, the modifications for the RVMCTS are tested against two of the validated benchmark rediscovery set described in Chapter 7. Finally, the modifications are combined to form REACTS (REACtion Tree Search) and solve both rediscovery problems.

Some "universal" modifications were introduced in Chapter 8, such as duplicate removal and limited depth. For this work, those additional universal modifications are maintained.

9.1.1 RVMCTS modifications

Modifications to the RVMCTS aim to ensure that the limitations identified in Chapter 7 and Chapter 8 are mitigated. The MCTS algorithm is a flexible modular algorithm separated into four components: selection, expansion, simulation, and backpropagation, and various modifications are applicable to each component. Broadly, MCTS modifications are split into two categories: domain-independent modifications; and domain-dependent modifications. Domain-independent modifications are methods designed to improve the quality of the tree search regardless of the problem type under investigation. Domain-dependent modifications seek to improve the quality of the tree search for one problem type. In both cases, a testing methodology needs to be outlined. As modifications to the tree search are generally non-additive, it is crucial that testing is carried out systematically. In this chapter, the modifications shown in Table 9-1 will be introduced, implemented, and tested. The modifications are categorised based on the part of the tree search which is modified:

Selection	Expansion	Simulation	Backpropagation
Max-SAUCT (Linear cooling)	Probabilistic RV pruning	None	Max Score backpropagation (Only with Max-SAUCT)
Max-SAUCT (Geometric cooling)	Physchem property pruning		
PUCT (Predictor UCT)	Local Chemical Subspace pruning		

Table 9-1 Investigated modifications to the RVMCTS categorised based on their function

9.2 Methods

9.2.1 Modified selection

Many different approaches can be utilised for selection in the MCTS. Some key considerations need to be taken into account when modifying selection in the RVMCTS implementation. First, information is backpropagated and, therefore, modification to selection also leads to modifications of backpropagation. Second, selection must allow for local optima to be gradually weakened as better routes in the tree are observed. Hence, the weakening of routes leads to previously reasonable solutions becoming less favoured if better solutions are discovered in other paths of the tree. Third and finally, the selection function needs to be fast to calculate so that the modified selection approach must not rely strongly upon updating expensive models.

The RVMCTS investigated in Chapter 7 used the UCT equation derived from the UCB for multi-armed bandit problems. The UCT equation was designed initially for scores of either zero or one (win/loss). The UCT equation is shown below:

The exploitation term is the average score of the subtree that grows beneath the node.

$$UCT = Exploitation + Exploration$$

$$UCT = Average\ of\ Subtree + Exploration\ term$$

$$UCT = \frac{Total\ Score\ of\ the\ node}{number\ of\ visits\ to\ the\ node} + 2 \times C_p \times \sqrt{\frac{2\ln(number\ of\ visits\ to\ parent)}{number\ of\ visits}}$$

Where:

$$Total\ Score = Score\ of\ the\ current\ state + \sum score\ of\ children$$

C_p is the exploration term that is used to tune the amount of exploration that occurs during the search.

As outlined previously in Chapter 6 and summarised here, selection needs to balance exploration and exploitation of the search space. However, it was clear in Chapter 7 that selection failed to work effectively across the three benchmark problems. Furthermore, the ablation study in Chapter 7 showed that when the selection component was removed, the algorithm performed better than with the UCT equation for two of the three problems. The UCT fails to solve rediscovery problems in reaction-based de novo design for two reasons: the exploitation term leads the search towards the node with the highest average score in its subtree rather than the maximum score; and the exploration term grows faster than the exploitation term.

Considering the first problem, using the average of the subtree for rediscovery problems assumes that a high average score will lead to the maximum of the search space being found. This assumption may not be valid when high-scoring nodes are hidden amongst siblings that are poor scoring. This issue of high average scoring paths surrounded by poor scoring paths is a well-known limitation of the MCTS algorithm (Bjornsson & Finnsson, 2009). An alternative way of framing the problem is that the RVMCTS will prefer conservative, safe routes in high average regions of the search space rather than high-risk high-reward regions of the search space.

The second issue arises due to the large number of molecules being generated during the search, such that the ratio of parent visits to child visits (a crucial part of the exploration term) increases faster than the average score of the subtree. This imbalance is because many child molecules are generated during expansion, which leads to the visit count for the parent molecule being large and that of the child molecule being low. This leads to a bias towards molecules that generate many children. However, even though C_p was set low to try and maintain a balance of exploration-exploitation in the ablation studies, the C_p parameter is difficult to tune and requires several runs to identify an optimal value.

Overall, it is challenging to build robust selection functions that balance exploration and exploitation across several different problem types. In this work, rediscovery problems are explored; however, further uses cases need to be considered when designing the selection function. The difficulty arises because de novo design has many problem types with very little correspondence between them. Furthermore, de novo design has an infinite supply of problem examples per problem type, and therefore this further exacerbates the difficulty of building a robust all-round selection modification.

Three modifications to selection that are investigated are described below: Max-SAUCT (Linear cooling); Max-SAUCT (Geometric cooling); and Predictor UCT.

9.2.1.1 Max-Simulated annealing UCT (Max - SAUCT)

Max-Simulated annealing UCT(Max-SAUCT) is a new selection function derived in this work explicitly to find the maximum of the subtree while simultaneously reducing the amount of exploration performed during selection. Max-SAUCT is inspired by SAUCT shown in equation (15)

. SAUCT was derived initially to improve balancing the exploration and exploitation of MCTS in a domain-independent context (Ruijl et al., 2014). SAUCT includes an additional T_i term which allows dynamic control of the amount of exploration. SAUCT allows for more exploration at the beginning of the search with the balance shifting to exploitation as the search progresses. This allows greater depth and higher overall scores to be achieved in the problem of mathematical expression creation (Ruijl et al., 2014). Max-SAUCT shown in equation (16) is a further extension of the SAUCT selection function to include the maximum of the subtree rather than the average of the subtree. The number of visits is also removed from the first term of the Max-SAUCT to force the search towards the maximum of the subtree. Furthermore, the second term has been simplified to reduce the impact of the number of children per parent.

$$SAUCT = \frac{Total\ Score\ of\ the\ node}{number\ of\ visits\ to\ the\ node} + T_i \times 2 \times C_p \times \sqrt{\frac{2\ln(number\ of\ visits\ to\ parent)}{number\ of\ visits}} \quad (15)$$

Where T_i is:

$$T_i = \frac{Total\ iterations - current\ iteration}{Total\ iterations}$$

C_p is the exploration term set as a constant at the beginning of the search.

Max-SAUCT is then given as:

$$\text{Max-SAUCT} = Max(subtree) + T_i \times \sqrt{\frac{\ln(number\ of\ visits\ to\ parent)}{number\ of\ visits}} \quad (16)$$

One issue with this approach is the rate at which cooling occurs during the search. Originally the authors of the SAUCT equation (Ruijl et al., 2014) proposed the linear cooling rate. However, different cooling schedules can be defined and have been extensively explored in the simulated annealing literature (Abramson et al., 1999). The linear and geometric cooling schedules that are used here are shown in Table 9-2.

Cooling Schedule	Formula
Linear	$T_i = \frac{Total\ iterations - current\ iteration}{Total\ iterations}$
Geometric(Exponential)	$T_i = \frac{Total\ iterations - current\ iteration}{Total\ iterations} \times \left(\frac{1}{\frac{Total\ iterations - current\ iteration}{Total\ iterations}} \right)^{\frac{Current\ iteration}{Total\ Iterations}}$

Table 9-2 Different cooling schedules investigated in Max-SAUCT

9.2.1.2 Predictor UCT (PUCT)

A different approach to selection is the Predictor-UCT (PUCT) (Rosin, 2011). PUCT has found popularity in the Monte Carlo tree search(MCTS) literature (Silver et al., 2016) as the primary approach for selection across many domains. PUCT uses the probability output from a model as an estimate of the strength of a given action. The probability is generated during expansion when a node is created and is based on the SM and RV combination used to generate the node. The probability is annotated to the node. When selection is performed, the annotated probability is then used as part of (17)

. The probability is not updated after its initial annotation to the node.:

$$PUCT = \frac{Total\ score\ of\ the\ node}{Number\ of\ visits\ to\ the\ node} + C_P * P(s,a) * \frac{\sqrt{number\ of\ visits\ to\ parent}}{1 + number\ of\ visits\ to\ child} \quad (17)$$

Where:

$$Total\ Score = Score\ of\ the\ current\ state + \sum score\ of\ children$$

C_p is the exploration constant (set to $\frac{1}{\sqrt{2}}$), $P(s,a)$ is the probability from the model when provided a given state (SM) and action (RV).

C_p is chosen as this value as it has been demonstrated to be a theoretical optimum value for C_p (Kocsis et al., 2006) Moreover, the value is used during several other implementations using PUCT(Segler et al., 2018; Silver et al., 2018).

The probability used by the PUCT is generated using a CB model described in section 9.2.2.3 below.

9.2.2 Modified Expansion – Pruning

Pruning is the process of removing nodes or edges from the tree. Pruning is essential as it speeds up the overall search by reducing the number of nodes present in the search tree (Birmingham & Kent, 1988). Two strategies can be used to reduce the number of nodes in the tree: reduce the number of actions that can be applied to a node (pre-pruning); reduce the number of states after application of the actions (post pruning). Furthermore, pruning can be hard or soft. A hard prune removes a node from the tree, and the pruned node will never be added back to the tree during the search. A soft

prune removes the node, but the removal is temporary, and the node can be re-introduced to the tree at a later stage (Browne et al., 2012). Hard pruning is used in this work.

The rationale for focusing on hard pruning is two-fold. First, the combinatorial explosion generated by the search is huge, as demonstrated in the ablation studies in Chapter 7. Second, soft pruning leads to further complexity in how nodes should be reintroduced to the tree. In the first case, the ablation studies of Chapter 7 demonstrated that ablation of expansion (random pruning) improved the overall computational time. Therefore, while reintroducing nodes at a later stage may be helpful, it would prevent deeper trees from being grown due to nodes having to be reintroduced. In the second case, soft pruning requires both a criterion for removing nodes in the tree and a criterion for adding nodes back to the tree; hence, although soft pruning is an exciting avenue to possibly improve the performance of the MCTS, the exceptionally high branching factor found in the RV search space and the additional algorithmic complexity led to the decision to focus on hard pruning.

Moreover, pruning is sometimes separated into safe and unsafe pruning. Unsafe pruning is where nodes are removed from the search tree without guaranteeing that they do not lead to good solutions. In contrast, safe pruning is when nodes are removed from the tree where it can be proved that they will not lead to good solutions. In most cases, pruning is unsafe.

The section below describes post pruning and pre-pruning methodologies.

9.2.2.1 Post pruning

The simplest approach to pruning is post pruning, which removes nodes or edges from the tree that have already been generated. Post pruning allows for a lot of flexibility as it allows for domain knowledge to be introduced to the search. Similar criteria can be used for post-pruning in de novo design as are used in virtual screening (VS). In VS, the goal is to screen an extensive catalogue of compounds to identify promising compounds based upon a predefined objective. One of the most common examples of ligand-based VS is negative design (Yang et al., 2020), which attempts to prioritise compounds based on removing compounds with undesired properties. Lipinski's rule is an example of negative design, where compounds are removed that are unlikely to be orally soluble and bioavailable (Lipinski et al., 1997). A further example of negative design is the use of substructural alerts (Senger et al., 2016), where substructures that are thought to cause unwanted toxic effects are removed.

In this work, two different post-pruning strategies are explored: PhysChem pruning and local chemical subspace pruning.

First, PhysChem pruning is based on a fusion of negative design rules (Yang et al., 2020). The basic principle of constructing this set of rules is to remove molecules that are unlikely to be promising drug candidates, for example, excessively heavy, flexible, or polar molecules. The fusion rule is based on maximum and minimum values of descriptors. The minimum bound of the compounds' descriptors generated during the search is based on fragment rules and Ro5 (Baell & Holloway, 2010; Kelder et al., 1999; Lipinski et al., 1997), and the upper bound is based on the Beyond rule of five rulesets (Doak et al., 2015). The physicochemical values selected are shown in Table 9-3 below.

Descriptor	Minimum	Maximum
Mass weight	150	1000
cLogP	-5	10
HBA	1	15
HBD	0	6
Rotatable bonds	0	20
Number of rings	1	No maximum
TPSA	60	250
Permitted Atom types	C, H, O, N, P, S, F, Cl, Br,I	

Table 9-3 PhysChem pruning maximum and minimum descriptor values

Second, local chemical subspace pruning (LCS) is based on locally pruning regions of chemical space. LCS approaches have been used in atom-based de novo design algorithms (A. Nigam et al., 2021; Verhellen & Van Den Abeele, 2020). These methodologies remove molecules from the pool of candidate compounds if they are not best in their local region of chemical space. To do this, an archive is formed, which is based on a subdivision of descriptor space. Each cell contains the score of the best compound in that local region of descriptor space. When a molecule is generated, if its score is greater than the best seen so far for that particular cell in the descriptor space, it is retained, otherwise, it is discarded. This simple local chemical subspace pruning rule has been shown to improve significantly the overall scores of generated compounds in genetic algorithm based de novo design and to improve the diversity of the generated population (A. Nigam et al., 2021; Verhellen & Van Den Abeele, 2020). A visual description of the process of using chemical subspaces to prune generated molecules is shown in Figure 9-1 below.

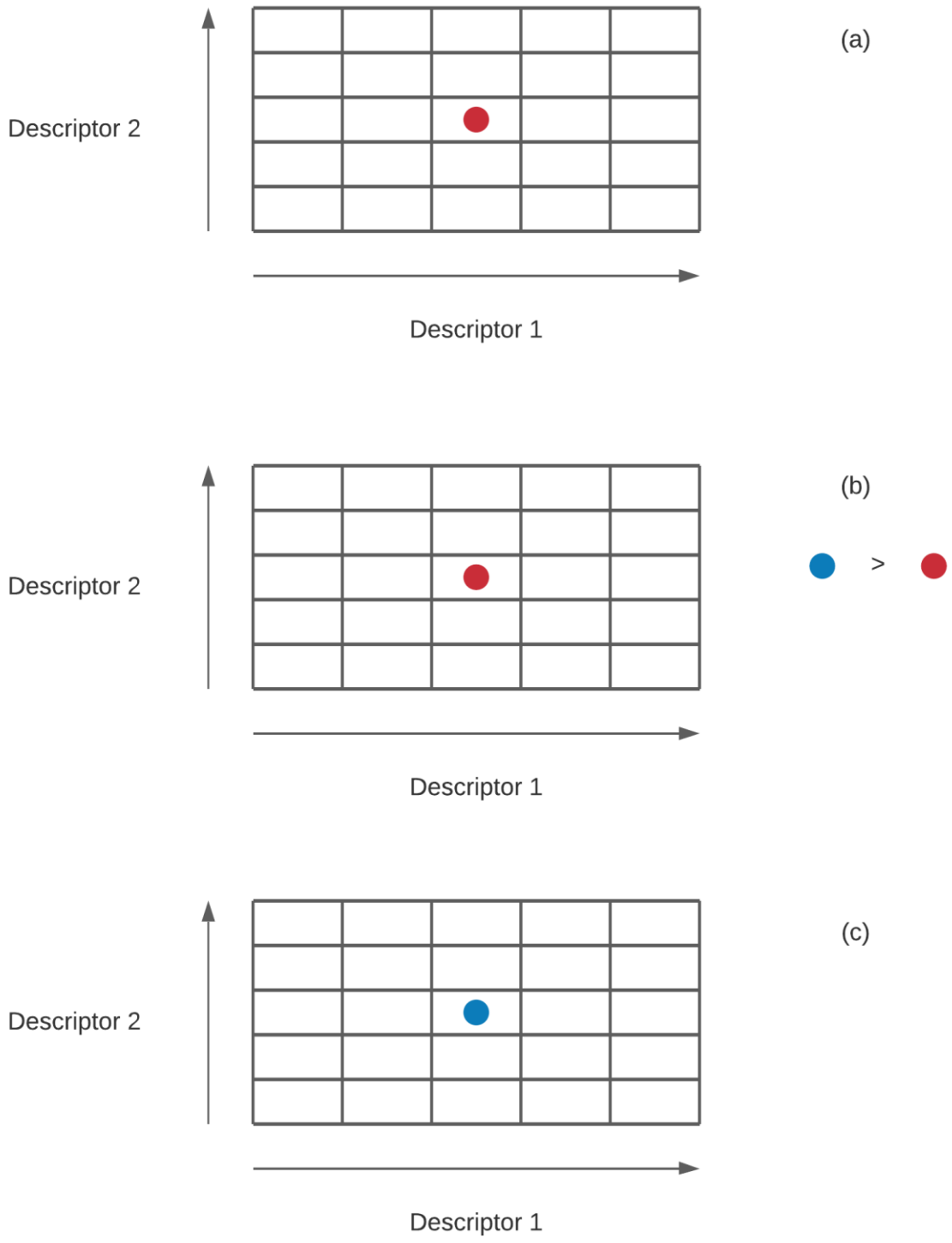


Figure 9-1 a) A region of descriptor space partitioned into 25 cells. The middle cell is filled with a red dot. The red dot signifies the highest scoring compound discovered so far in the search. b) a new compound has been generated (blue dot), and its score is higher than the red dot. Therefore, the blue dot will replace the red dot.c) the blue dot has replaced the red dot as the best in the cell.

Integrating this process into the search is relatively straightforward. At each expansion, each molecule generated is checked against their corresponding cells. If the generated molecule is the best in that cell, it will be retained in the tree, otherwise it is discarded. Pruning only occurs during each expansion phase so that molecules generated in previous iterations are not removed.

The partitioning for the LCS pruning approach is shown in Table 9-4 below.

Descriptor	Minimum Value	Maximum Value	Number of partitions
Molecular weight	140	520	20
LogP	-0.6	5.6	20

Table 9-4 Partitions used during the local chemical subspace pruning

9.2.2.2 Pre-pruning

Reaction-based de novo design aims to design molecules with predefined properties while proposing possible synthetic routes. Therefore, in reaction-based de novo design, pre-pruning can be based on both synthetic feasibility and the possibility of whether the compound will lead to high scoring molecules.

In the RVMCTS implementation, pre-pruning based on reaction feasibility occurs currently via the RV selection process (Hristozov et al., 2011; Patel et al., 2009; Wallace, 2016). From Chapter 7, it was seen that while the selection procedure used in the RV engine is a promising start, the number of applicable RVs is still too high to search through. Previous work has sought to use multi-label classification (MLC) based on SM - reaction class combinations seen in the literature to propose preferred reaction classes (Ghiandoni et al., 2020). The retrieval of RVs is then limited to the proposed reaction classes only. A practical limitation of this approach, however, is that If the model does not suggest a reaction class, then all RVs would have to be applied. In the RVMCTS, it is a certainty that during the search, there will be molecules generated that are outside the domain of applicability of the model, which will lead to the tree not being pruned as desired.

Figure 9-2 below shows a high-level overview of how the RV selection operates. Here the SM is first passed into the RV select code, which returns the list of applicable RVs. Then, for each applicable RV, appropriate reagents are found. There are two important points here: 1) the applicable RVs must always be determined before the reagents; and 2) the number of applicable reagents will likely be greater than the total number of applicable RVs.

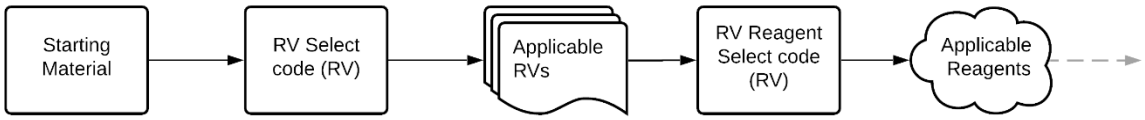


Figure 9-2 RV selection code without pruning

A probabilistic RV pruning process has been integrated into the RVMCTS as shown below. Probabilistic RV pruning is a pre-pruning method applied to RVs that uses a probabilistic CB model to rank the applicable RVs from which the top-ranking RVs can be selected. A description of probabilistic RV pruning is provided in Figure 9-3 below.

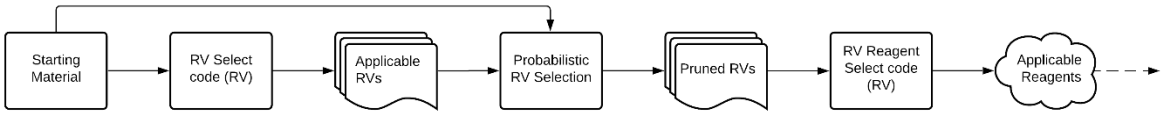


Figure 9-3 RV selection code with pruning

9.2.2.3 Probabilistic RV Pruning

As highlighted previously, the number of applicable RVs is too high, leading to poor overall performance, therefore pruning of the list of applicable RVs needs to occur. To prune the list of applicable RVs, the outcome of applying each RV to a SM needs to be predicted. The outcome of applying an RV to the SM is referred to as its reward and is the score of the product molecule (or the average score when more than one product is generated). Therefore, to prune the list of applicable RVs, a method of predicting the reward for each unique SM and RV combination needs to be determined (the predicted reward is referred to as the expected reward).

In probabilistic RV pruning, the expected reward for each applicable RV is determined and these are converted to a set of probabilities which are then used to prune the RVs. This is achieved using a CB model. A CB model is a reinforcement learning algorithm that takes a list of SM and RV combinations and returns a list of probabilities, one for each RV. The higher the probability, the higher is the expected reward the CB model believes can be achieved by that specific RV.

A CB model relies on two parts: a learner and an exploration algorithm. The learner maps a set of inputs to the set of expected rewards, and the exploration algorithm maps the set of expected rewards to the set of probabilities. In this context, inputs are the list of SM and RV combinations, and the outputs are the list of probabilities in the probability function. To prune the list of applicable RVs, the predicted probabilities are sorted highest to lowest, and the top k probabilities are selected.

The inputs and the outputs of the CB model are shown in Figure 9-4 below.

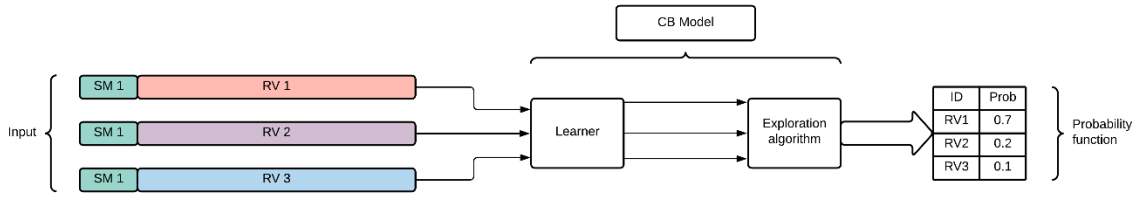


Figure 9-4 outline of a CB model inputs are passed into the Learner which predicts a series of values. These values are then passed to an exploration algorithm which converts those predictions into a probability function.

In this work, the learner is a linear regression model, and the exploration algorithm is a softmax lambda function.

A critical feature that separates the CB model from a standard regression model is the process of training. Training of the CB model uses a process of reinforcement learning. Reinforcement learning is where the model is given a SM and a list of applicable RVs, and the CB model predicts a set of probabilities one for each applicable RV. In the first pass of the CB learner the rewards will be assigned at random. The rewards are then converted to probabilities. Then, a single applicable RV is chosen by sampling the set of predicted probabilities. The true reward is calculated (by generating and scoring the product molecule(s)) and is then concatenated with the input (SM and RV) and the predicted probabilities. The annotated input is then given to the CB model, which updates both the learner and exploration algorithm by stochastic gradient descent. As a result, the learner will minimise the difference between the expected reward and the true reward. The exploration algorithm will also update its ability to predict probabilities.

One CB model is trained for each rediscovery problem. To train the CB model, a dataset of SMs and lists of applicable RVs are required. The dataset of SM and lists of applied RVs that were generated for the work in Chapter 8 were used here. Additionally, during reinforcement learning, rewards need to be provided for the model to learn. The reward for an SM and RV combination is the geometric mean of the similarities of the product molecules to the goal molecule. In general, multiple products are generated as there are multiple reagents that can be used. Figure 9-5 below illustrates the process of computing the geometric mean of the similarities to the goal molecule for a unique SM and RV. The process leads to a single SM and RV having a single value annotated which is used as the true reward during the reinforcement learning training process.

The process of generating the dataset yielded 6.05M unique SM and RV combinations, as shown in Figure 9-6. Finally, the SMs and RVs need to be turned into inputs that the CB model can read before training. The inputs were the RV fingerprint and the Morgan fingerprint (16384 bits, radius=2).

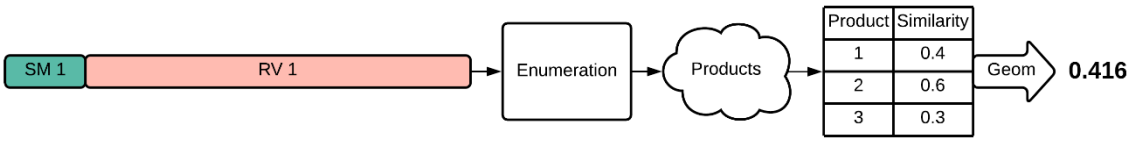


Figure 9-5 SM one (SM 1) and RV one (RV 1) are enumerated and generate a set of three products. Each product is annotated with a similarity to the target. The geometric means of the similarities to the target are then determined (0.416).

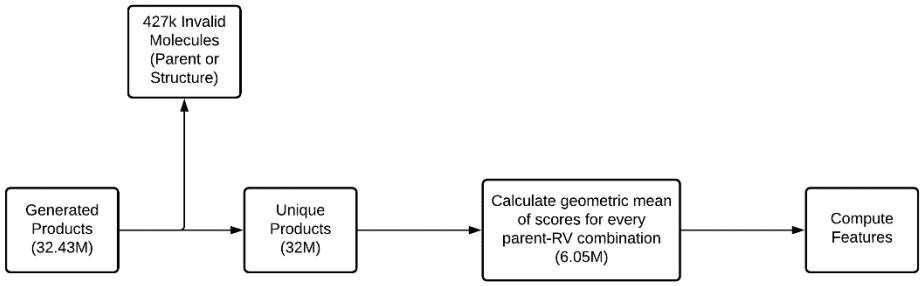


Figure 9-6 dataset generation

9.2.2.3.1 Training the CB Model

In contrast to the regression modelling problem in Chapter 8, training of the CB models occurs via reinforcement learning.

The process of training a CB model is as follows:

1) Randomly choose a single SM and corresponding list of applicable RVs from the dataset (Figure 9-6)
2) For each RV in the list of retrieved RVs, run the following subprocess to generate the set of inputs:
    a. Concatenate the RV fingerprint and Morgan fingerprint (16384 bits, radius =2) of the SM.
3) For the list of SM and RV combinations, predict the set of probabilities which consists of one probability for each SM and RV combination.
4) Sample the set of probabilities to select a single SM and RV.
5) Retrieve the reward for the SM and RV and concatenate with the input and predicted probability to form a training example
6) Update the CB Model using stochastic gradient descent, minimising the error between the predicted and true reward.
7) Repeat steps 1-6 until a predefined number of training examples have been learnt.

An example of one iteration of Steps 1-6 is shown in Figure 9-7. In this work, each model was trained for 1M iterations. During training, the difference between the predicted (expected) reward and the true reward is minimised. By decreasing the difference between the predicted reward and the true reward, the corresponding exploration algorithm can make better-predicted probabilities.

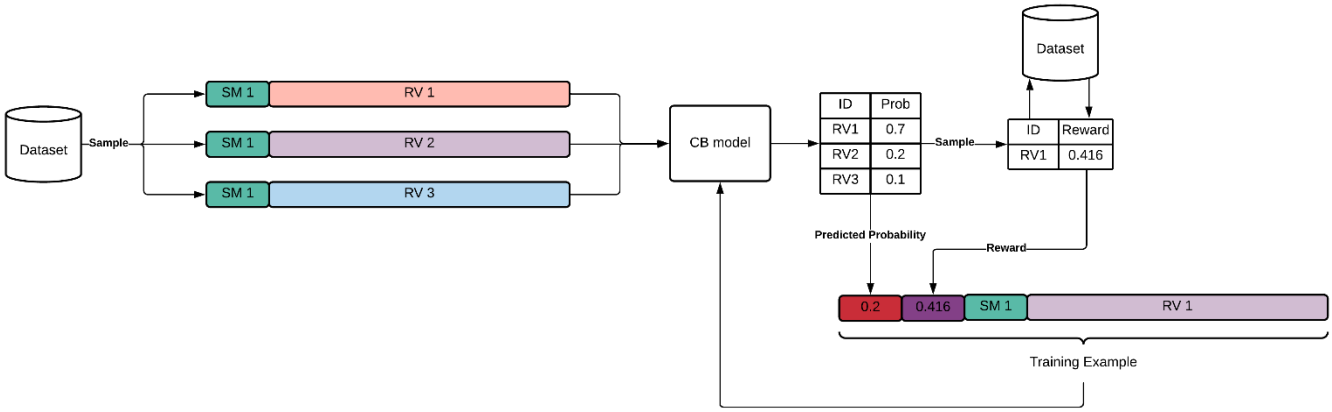


Figure 9-7 example of a single update of the CB model.

To train each CB model for 1M iterations took approximately nine hours on the Sheffield HPC facility.

9.2.2.3.2 Validation of the CB models

One difficulty of training a model using reinforcement learning is validation. The validation of CBs is a complex topic (Blum et al., 1999) with the complexity stemming from the method of sampling used to train the CB model (See Figure 9-7).

Therefore, to understand if the CB models have learnt to predict better probabilities, the average predicted reward could be compared with the average true reward. Hence, If the mean observed true reward during training is higher than the average true reward of the dataset, then the probabilities being generated by the model are favouring RVs with high rewards and thus would be sampled more often during the training process. This is equivalent to comparing the ability of the CB model to predict high scoring RVs against randomly choosing RVs from the dataset.

Table 9-5 below shows the final model performance after training with the corresponding average expected value for the dataset.

Model	Mean reward	Mean reward CB model
Aripiprazole	0.107	***0.110***
Ambrisentan	0.131	***0.134***

Table 9-5 Final performance of the trained CB models

The CB models were also validated using the rankings of a holdout set. A holdout dataset of SMs and lists of applicable RVs was generated. The SMs were 3k molecules additionally sampled from the ZINC20-in-stock dataset used in the regression modelling task in Chapter 8. Care was taken to avoid selecting SMs used in the training of the CB model. The 3k sampled molecules were expanded and processed using the same methodology described in Figure 9-5 and Figure 9-6. Following processing, this generated a dataset comprising 2106 SMs with corresponding applicable RVs and lists of geometric means of the similarities to the goal molecule (one for each applied RV in the corresponding list of applicable RVs). The RVs for each SM were ranked based on the geometric means. Finally, for each SM and corresponding RV, the inputs (MorganFP+RVFP) were created, and the CB model was used to predict the set of probabilities for each SM and list of applicable RVs and the probabilities were ranked.

The true rankings were then compared with the predicted ranking generated by the CB model using the Normalised Discounted Cumulative Gain (NDCG) score (Järvelin & Kekäläinen, 2002). An NDCG score of 0 means the rankings are inverted (the highest geometric score would be given the lowest predicted rank). An NDCG score of one means the rankings are perfectly matched. The NDCG scores were averaged across the dataset.

The NDCG score can also be calculated for different lengths of lists of applicable RVs, such as the top k applicable RVs. The NDCG score is then referred to as the NDCG@K. NDCG@5, NDCG@10, NDCG@15 and NDCG@20 were computed and then averaged across the dataset.

The Mean Reciprocal Rank (MRR) was used to measure the position of the true highest-scoring RV in the ranked list of predicted probabilities (Voorhees & Tice, 2000). For example, if the highest-scoring RV has a position of one in the ranked list of probabilities, then the reciprocal rank score is one. If the highest-scoring RV has a position of two in the ranked list of probabilities, the reciprocal rank score is ½. The MRR is the average position of the highest-ranked RV in the ranked list of predicted probabilities computed across all lists in the dataset.

The results of the validation are shown in Table 9-6 below. Promisingly, both trained CB models show good ranking ability due to their high NDCG score and good MRR performance.

Model	NDCG	NDCG@5	NDCG@10	NDCG@15	NDCG@20	MRR
Aripiprazole(CB)	0.938	0.871	0.874	0.887	0.896	0.681
Ambrisentan(CB)	0.943	0.878	0.886	0.883	0.896	0.613

Table 9-6 Validated NDCG@k scores and MRR for the trained CB models.

The models are integrated into the tree search during the expansion phase. The model is supplied with a list of applicable RVs, and the k most probable RVs are applied.

9.3 Experimental

The performance of the modifications was tested on two of the rediscovery problems (aripiprazole and ambrisentan). The performance of the modifications was tested on two of the rediscovery problems (aripiprazole and ambrisentan). The rationale for focusing on only these rediscovery problems was that the alogliptin problem was solved using a greedy search method (ablation of simulation) and implies that the alogliptin search space is more accessible to traverse than the other two rediscovery problems. Hence, the experimental focus shifted to the two perceived more challenging search problems. The second reason is that the computational cost of testing modifications was high. Therefore, the decision to focus on the two more complex problems was reinforced to allow faster experimental turnover.

The following search configurations were used with one modification per configuration;

1) Ablated simulation (Benchmark)
2) Max-SAUCT (Linear cooling)
3) Max-SAUCT (Geometric cooling)
4) PUCT (No pruning probabilities only)
5) PhysChem pruning
6) Local chemical subspace pruning
7) RV Prune (k=1)
8) RV Prune (k=5)
9) RV Prune (k=10)
10) RV Prune (k=15)
11) RV Prune (k=20)

The RVMCTS parameters for each search are as shown in Table 9-7.

Parameter	Value
Iterations	50
Exploration constant (C_p)	UCT (0.001)
RV Database	JMedChem 4.5k + seeded RVs
Reagents	Internal reagents
Simulation	None
Backpropagation	Continuous
Depth	Aripiprazole (3), Ambrisentan(4)

Table 9-7 RVMCTS parameters for the modification experiments

The total number of replicates per search was set to five. The searches are compared using the same metrics as shown in Chapter 8.

9.4 Results

9.4.1 Maximum Score

Figure 9-8 shows the maximum score found in the trees for the different search configurations. The first modifications were to selection and were designed to improve the balance of exploration and exploitation. In the aripiprazole problem Max-SAUCT (Linear cooling) and Max-SAUCT (Geometric cooling) failed to find the goal compound, however, in both cases a molecule was found with a high overall score (>0.7 Tc). In the ambrisentan problem, both of the simulated annealing selection approaches solved the ambrisentan problem. This suggests that for ambrisentan the difficulty for the search was a lack of exploration and the simulated annealing selection methodologies overcame this difficulty. PUCT also performed well, however the reduced performance on the ambrisentan problem is likely due to the limited number of iterations and if these were increased the goal molecule may have been found.

The next set of modifications were post-pruning filters. Post pruning removes nodes after they have been generated. Post pruning filters here were a PhysChem filter and a LCS filter. In the aripiprazole problem, the PhysChem and LCS filters provided no discernible benefit to the overall maximum score. In the ambrisentan problem, the LCS filter improved the maximum score slightly, however the maximum score was still poor.

Finally, the last modification was RV pruning. In the aripiprazole problem, RV Prune (k=5) successfully solved the aripiprazole problem, however, this was not the case for the other values of k (1, 10, 15 and 20). In the case of RV prune (k=1) this is likely due to being trapped in a dead-end molecule. In the case of RV prune (k=10, 15 and 20) the poorer performance is likely due to there being insufficient iterations. In the ambrisentan problem, all values of k returned the same result.

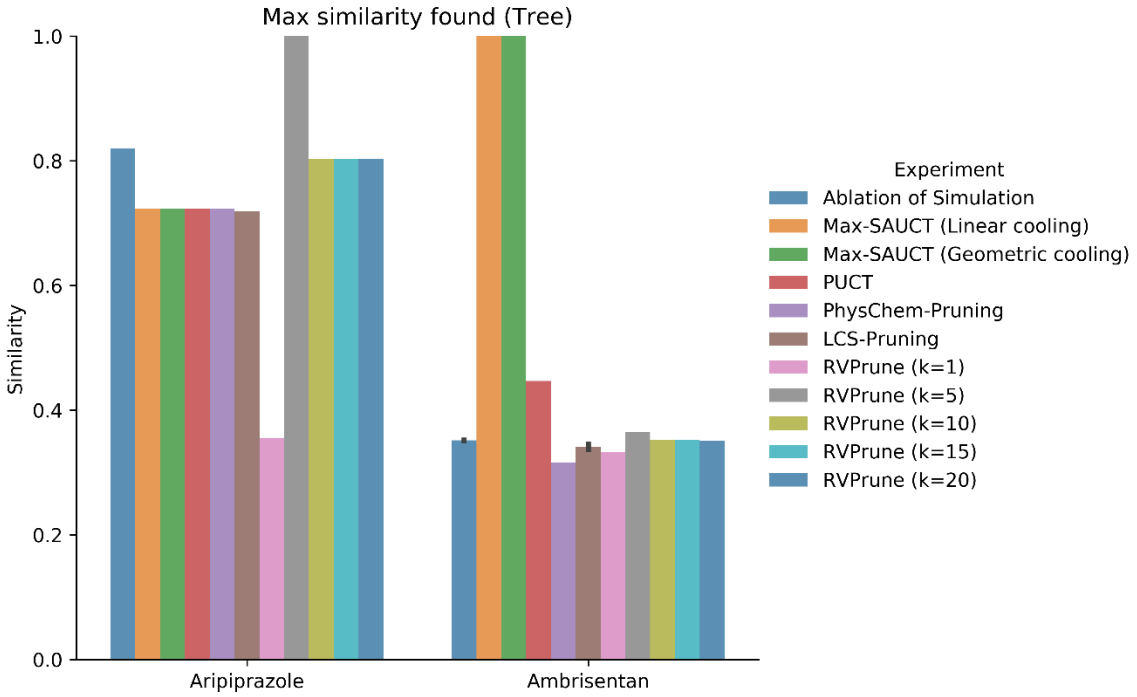


Figure 9-8 Maximum similarity found. The height of the bar is the mean, and the black bars are ± 1 standard deviation. No black bar represents a standard deviation of zero.

9.4.2 Time

Time is the total time required for the search to complete. Time has been shown previously to be directly related to the number of molecules generated. Fewer molecules generated mean less time spent generating new molecules in future expansions and less time spent scoring molecules. Figure 9-9 shows the total time spent in minutes for each of the searches.

Firstly, the Max-SAUCT (Linear cooling) and Max-SAUCT (Geometric cooling) slightly increased the overall time taken for the search to complete compared to the ablation of simulation baseline. The increased time taken is likely due to the avoidance of dead-end molecules due to wider exploration. In contrast, this was not observed in the ambrisentan problem due to fewer dead-ends being present in the tree. In the PUCT searches, there was a similar overall trend.

In both pruning approaches (pre- and post-) contrasting results are observed across each problem and therefore it is hard to draw general conclusions. In the aripiprazole LCS search, there was a large decrease in the overall time taken, this was probably due to the LCS-Filter removing molecules that were previously being expanded. In contrast the ambrisentan problem which did not benefit from the LCS pruning.

Finally, the modified RV prune saw a dramatic decrease in the overall number of molecules generated and thus a substantial decrease in the overall time taken. Interestingly, the different values of k for RV prune did not result in any difference in the overall time taken for the search to complete for aripiprazole. This is likely due to the presence of dead-end molecules which were generated and which the search gravitated towards. In contrast, in the ambrisentan problem there was a smooth increase in the total time taken as the amount of pruning was decreased, suggesting that no dead-end molecules are present. The pruning results suggests that pruning can sometimes have unexpected influences on the growth of the tree which are hard to predict prior to the search occurring.

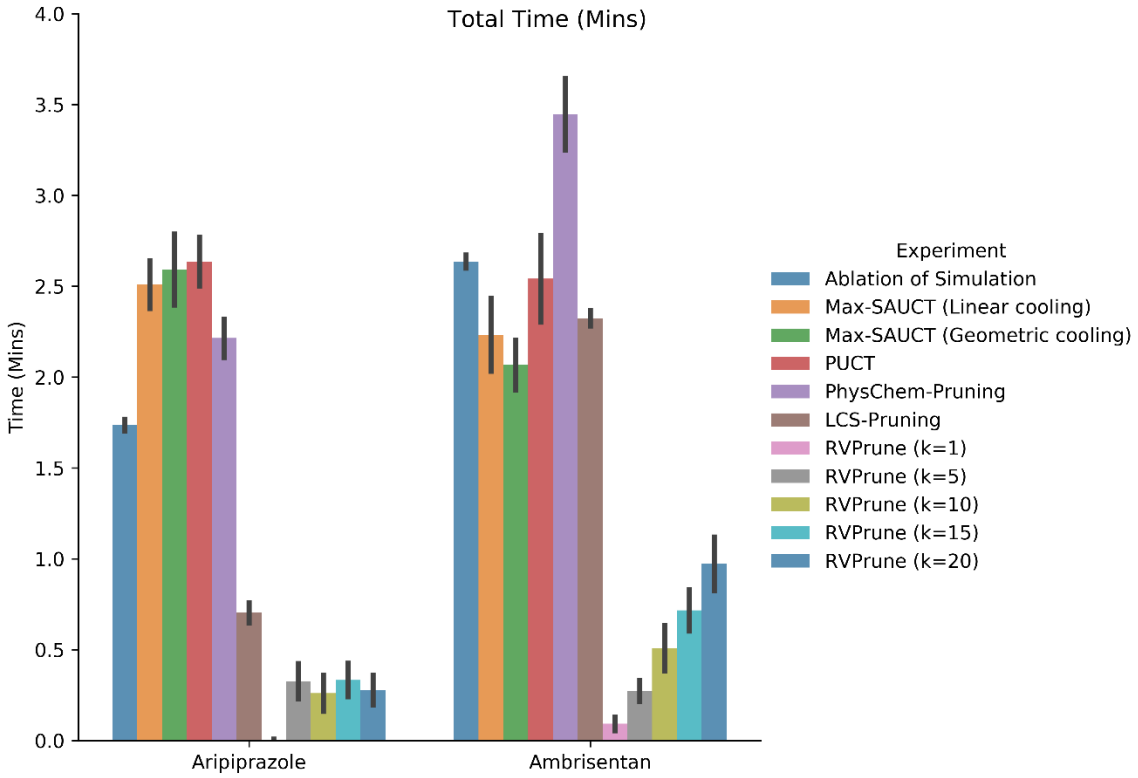


Figure 9-9 Total time (mins) The height of the bar is the mean, and the black bars are ± 1 standard deviation. No black bar represents a standard deviation of zero.

9.4.3 Average Score

The average score of the trees is a measure of the average quality of the search. Figure 9-10 shows the average performance of each of the searches. Starting with the selection modifications there was no discernible improvement using a modified Max-SAUCT variant across both problems. This is fairly intuitive as the maximum of the subtree is driving the search rather than the mean. Therefore, there is no reason why there should be a large improvement in the average score present within the tree. In the PUCT approach there was a slight decrease in the overall average during the search in the aripiprazole problem. This is likely due to the search exploring more and thus generating poorer scoring compounds which is consistent with the total time results. In the pruning approaches, there was an increase in the average observed for both PhysChem and LCS pruning in the aripiprazole problem compared to the selection modifications. This increase is because poor scoring compounds have been removed which has left, on average, higher scoring compounds in the tree. Interestingly this increased average was only observed in the ambrisentan problem for the LCS pruning.

Finally, the process of pruning RVs leads to an increase in the overall average score of the tree and then a steady decrease as less pruning occurs. This is again an intuitive result, as the CB model is designed to suggest high scoring RVs first and therefore, as fewer RVs are pruned the average would be expected to decrease. This is observed in the ambrisentan problem where there is a consistent decrease in the average score as fewer RVs are removed.

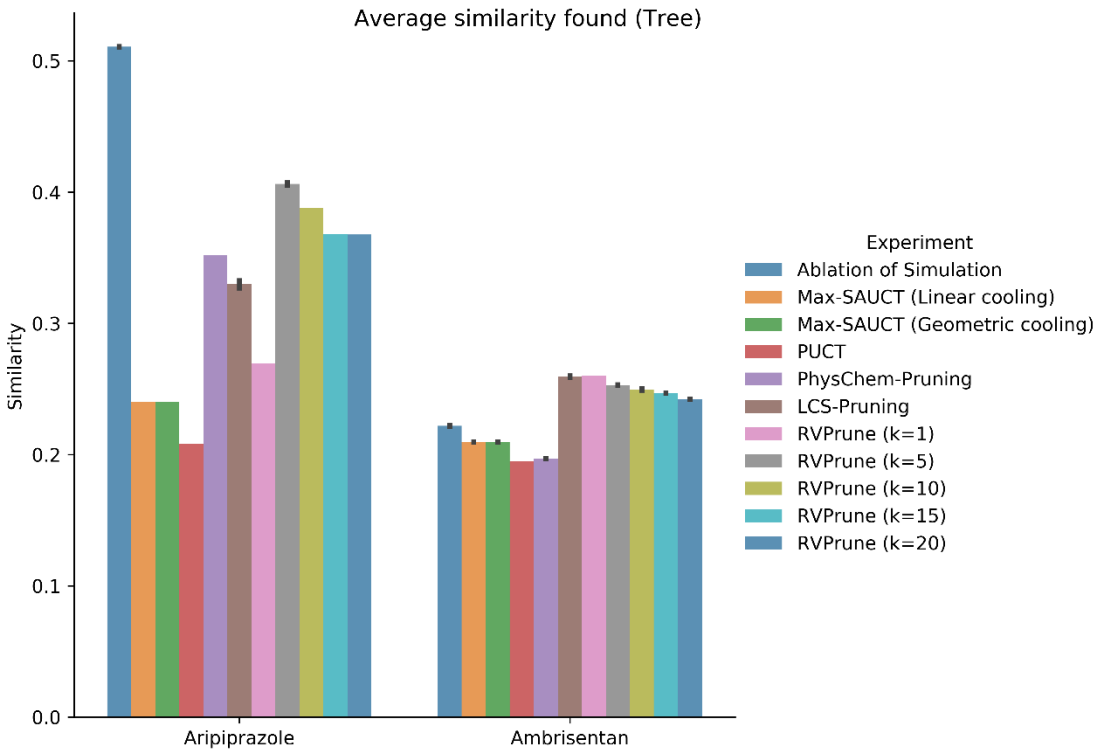


Figure 9-10 The height of the bar is the mean, and the black bars are ± 1 standard deviation. No black bar represents a standard deviation of zero.

9.4.4 Branching factor

Branching factor is a measure of the number of child molecules generated per parent compound. Generally high branching factors lead to poor performance of the overall search as many more compounds will need to be investigated. Figure 9-11 shows the overall branching factor of each of the modified searches. The modified selection approaches yielded a higher branching factor than the ablation of simulation baseline for the aripiprazole problem. In contrast, the branching factor was reduced in the ambrisentan problem. The lower branching factor for ambrisentan is likely due to increased exploitation as the search progresses compared to the case of the ablation of simulation where the balance between exploitation and exploration is constant throughput the search. This had the opposite effect for aripiprazole where the tree was wider. The same effect was observed in the PUCT searches as well with the aripiprazole becoming wider and the ambrisentan tree becoming narrower. In the PUCT this is likely due to the better overall balance between exploration and exploitation as well as the initial strength probability being supplied from the CB model. In the post filtering approaches, promisingly the branching factor was reduced for both problems. As molecules have been removed from the tree the overall branching factor has decreased. This would suggest that well parametrised pruning can lead to better searches as the branching factor could be reduced to a manageable level. Finally, RV prune yielded increased branching factors as fewer RVs were pruned.

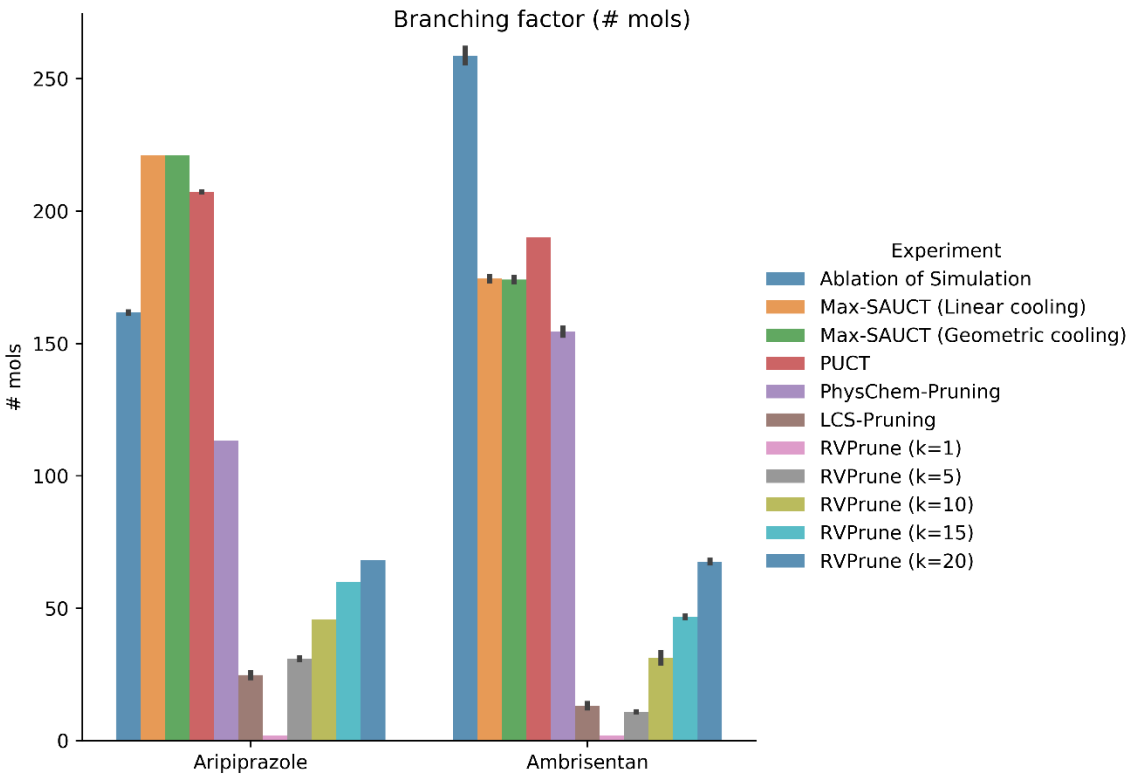


Figure 9-11 Branching factor. The height of the bar is the mean, and the black bars are ± 1 standard deviation. No black bar represents a standard deviation of zero.

9.4.5 Skewness of scores

Skewness is a measure of the direction of the tales of a distribution. In reaction-based de novo design a negative skewness is preferred as it suggests that the scores have been generated closer to the maximum value. The simulated annealing selection functions yielded skewness which was positive in both cases suggesting that a selection function which is focused on maximum score will not shift the entire distribution towards the goal. In constant, the PUCT equation generated lower skewness values which is promising and would suggest that if the search was given more iterations the distribution would shift considerably. In the pruning modifications there was a general trend towards shifting the skewness towards a negative value. In the aripiprazole example, the PhysChem pruning shifted the skewness more than the corresponding LCS pruning. In the ambrisentan problem this was reversed with the LCS pruning leading to a greater skewness than the corresponding PhysChem pruning. Pruning is likely to improve the skewness as weak performing compounds will be removed from the search. However, these results suggest that post pruning filters need to be parameterised to gain the maximum benefit during the search. Finally, the RV prune was the most successful approach in forcing a shift in the skewness of the score distribution for ambrisentan. This is intuitive as RVs are prioritised based upon their expected reward and therefore it is likely that those applied RVs will shift the distribution towards the goal.

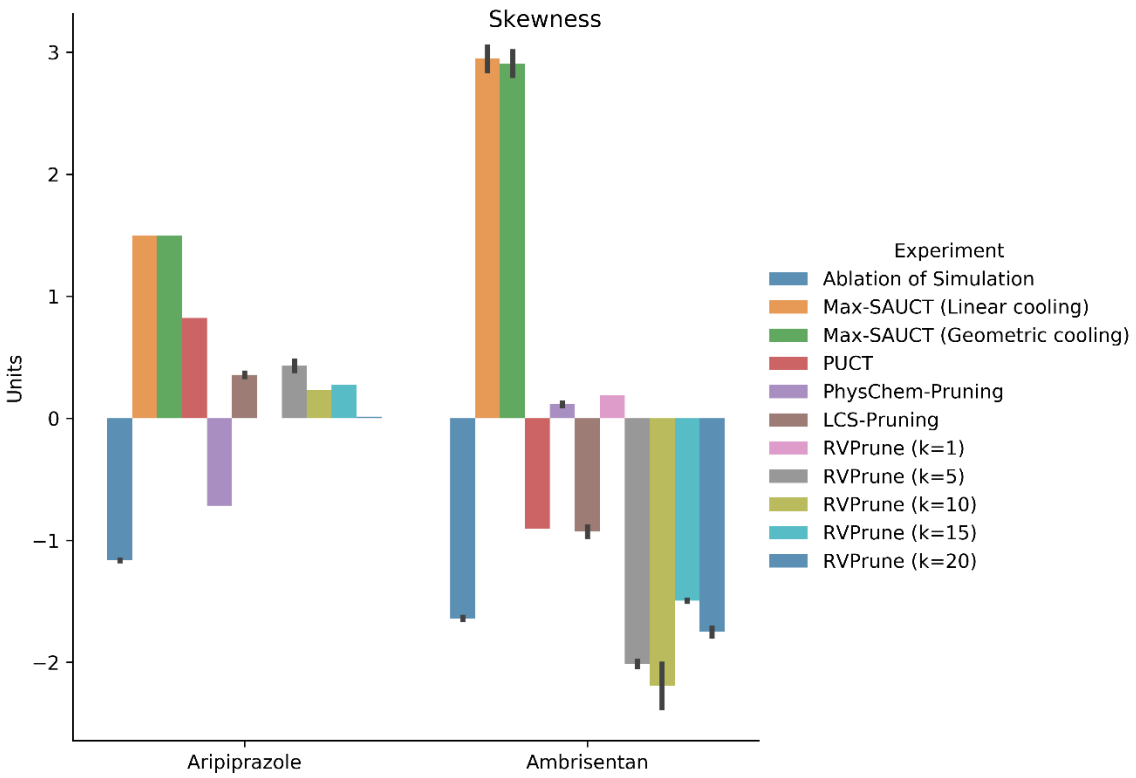


Figure 9-12 Skewness The height of the bar is the mean, and the black bars are ± 1 standard deviation. No black bar represents a standard deviation of zero.

9.4.6 Depth

Depth is the maximum number of reaction steps which the search reached. Previous approaches, as highlighted in the ablation studies chapter, failed to reach the maximum depth in both the aripiprazole and ambrisentan problems. Figure 9-13 shows the maximum depth reached during the modified searches. In the ablation of simulation baseline the goal depth is not capped. Promisingly in the aripiprazole case, in the majority of the modified searches the required depth was reached with the exception of PUCT and RV prune (k=1). PUCT not reaching the required depth is likely due to the wider branching factor and a focus towards using the average of the subtree. In the ambrisentan problem all searches except PUCT, PhysChem and RV prune (k=20) reached the required depth. It is likely that the PhysChem filters pruned away compounds that may have been high scoring and therefore this forced the search to explore a wider rather than a deeper tree which is desired. PUCT had a larger amount of exploration and thus did not reach the required depth. RV prune (k=20) yielded searches that had too many compounds and thus greater depths could not be reached.

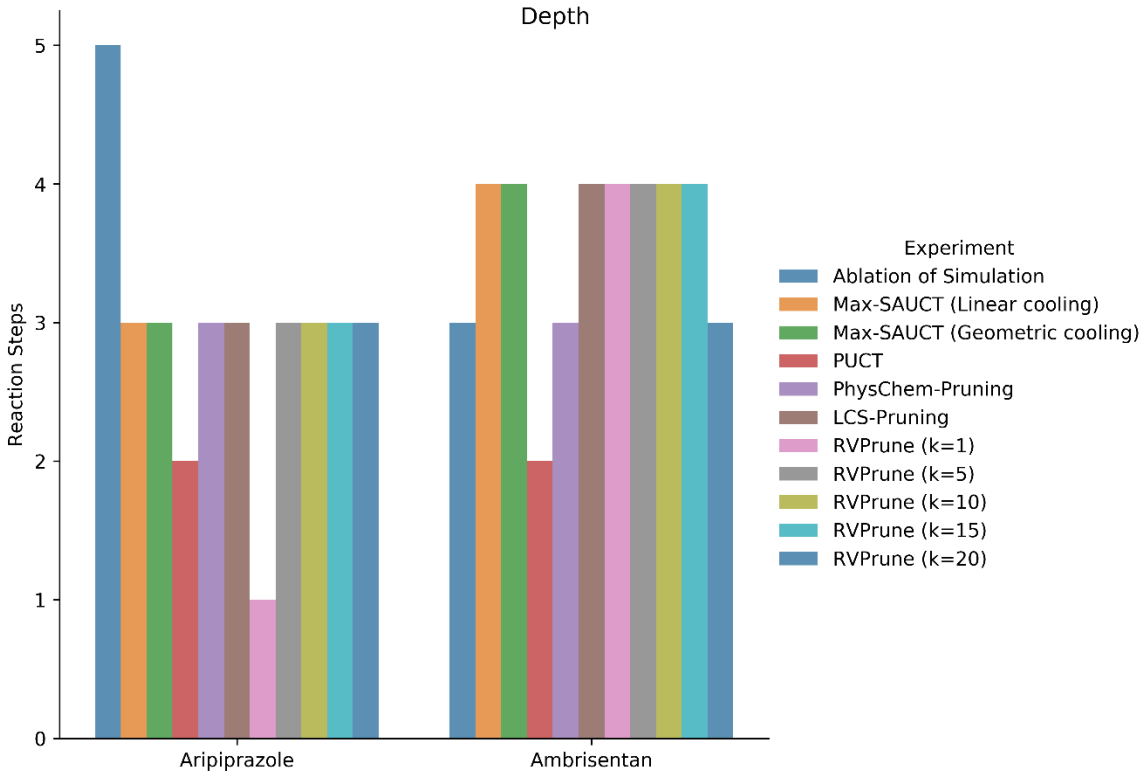


Figure 9-13 Depth The height of the bar is the mean, and the black bars are ± 1 standard deviation. No black bar represents a standard deviation of zero

9.5 Combining Max-SAUCT and PUCT

The maximum score and the average score results gave two contrasting results for the Max-SAUCT and PUCT selection methods. Firstly, the Max-SAUCT selection approach solved the ambrisentan problem, but did not solve the aripiprazole problem. Secondly, the skewness of the distribution would suggest that the Max-SAUCT is focussing on one path as the T value decreases. In contrast the PUCT algorithm performed relatively well but suffered from over exploration. To overcome these issues a SA component has been added to the second term of the PUCT equation. The new proposed selection function is called SAPUCT. The addition of the SA term allows the amount of exploration to decrease (improving exploitation at the end of the search).

$$SAPUCT = \frac{Total\ score\ of\ the\ node}{Number\ of\ visits\ to\ the\ node} + C_P * T_i * P(s,a) * \frac{\sqrt{number\ of\ visits\ to\ parent}}{1 + number\ of\ visits\ to\ child} \quad (18)$$

Where the definitions are the same as equation (17)

. This approach is preferred over Max-SAUCT (Linear cooling) and PUCT for several reasons. First, Max-SAUCT, can solve hard de novo design exploration problems (ambrisentan). However, this benefit came about since the path to solve ambrisentan (after the first step) continually generates the best compound seen so far. In contrast the aripiprazole problem generated an intermediate at reaction step two, which was higher scoring than the correct intermediate, hence even though Max-SAUCT solved the ambrisentan problem it's unlikely that it would be able to generalise to other de novo design problems. Second, Max-SAUCT does not contain any methodology to insert chemical information into the selection function. This means that the Max-SAUCT selection function cannot be tuned per problem. Third, the balance between exploration and exploitation in PUCT is constant, this again means that as the search progresses there is no shift towards exploitation. This limitation means that PUCT would maintain the same amount of exploration at the beginning of the search as the end even though many more molecules have been generated and added to the tree. Finally, PUCT, includes chemical information via the CB model, if the model was improved it would allow for better inclusion of domain specific information.

The SAPUCT improves on PUCT by allowing for exploration to slowly reduce over iterations. SAPUCT is used as part of the REACTS (REAction Tree Search) algorithm described below.

9.6 Solving the aripiprazole and ambrisentan problems

In the first part of this chapter, several experiments were carried out to understand the effectiveness of the proposed modifications. These experiments were based on considering each modification in isolation. In this section, different modifications are combined into a new algorithm

called REACTS with the choice of modifications informed by the findings reported above. REACTS also draws on findings from Chapter 8 and the machine learning approach that was introduced in the latter part of that chapter as a replacement for simulation.

The REACTS algorithm is described first, and its performance is then demonstrated on the two rediscovery problems. The algorithm is named REACTS (REAction Tree Search) and retains many of the components already discussed in the context of tree searching.

9.6.1 Methods

When combining different modifications care needs be taken. This is because combinations of different modifications can produce unintended side effects which may lead to poor performance. While performance is the most important metric to determine success of a search approach several other considerations were taken into account such as speed, generalisability, ease of use. Speed is important as it is likely that REACTS will be run for many iterations, if a single iteration takes several hours, then it would be a computationally unfeasible algorithm. Generalisability is a key consideration as ideally REACTS will be able to solve a variety of de novo design problems and therefore problem specific modifications should be avoided. Finally, ease of use is the final consideration as ideally a de novo design algorithm should require little parametrisation to deliver compounds with reasonable scores.

The set of modifications used in REACTS are the following

Component	**Modification**
Selection	SAPUCT
Expansion	RV prune, LCS pruning
Simulation	ML Mean simulation
Backpropagation	Continuous

Table 9-8 Modifications used by REACTS

SAPUCT was chosen for the selection function as it is takes positive elements from both Max-SAUCT (Linear cooling) and PUCT as described above. SAPUCT relies on C_p, and this has been set at $\frac{1}{\sqrt{2}}$ as for PUCT. Given that SAUCT includes a cooling factor that tunes exploration and exploitation as the search progresses, it is likely that more benefits will be seen for increased iteration counts. Therefore, the number of iterations is increase in the experiments describe below from the number used in previous experiments. Two expansion modifications are used. The first modification is RV prune. RV prune demonstrated exemplary performance across both problems with high maximum scores being achieved. Moreover, RV prune substantially reduced the time taken for the searches to

complete as fewer molecules were generated. Finally, the average score of the molecules present in the tree was improved for RV prune in both cases. The second expansion modification is LCS pruning. LCS pruning was chosen as it showed good reduction in tree size in the aripiprazole problem. LCS pruning is also a good choice as it allows intermediates to be maintained in the tree even if they have low scores. Finally, simulation is replaced with a ML Mean Simulation function as described in Chapter 8. The ML Mean Simulation function performed just as well as the regular depth one full enumeration but requires much less time.

9.6.1.1 Experiments

The number of iterations was increased to 500. The increase in the computational budget reflects a real-world scenario where a large number of iterations is used rather than a constrained search budget (Bradshaw et al., 2020; Button et al., 2019; Gottipati et al., 2020). Secondly, k=10 was chosen for RV prune as it appears to be a reasonable balance between the total branching factor and the maximum score discovered.

9.6.1.2 REACTS and LCS parameters

The REACTS parameters for each search used in this study are shown in Table 9-9 below:

Parameter	**Value**
Iterations	500
Exploration constant (C_p)	0.7071
RV database	JMedChem 4.5k + seeded RVs
Reagents	Internal reagents
RV prune	k=10
Depth constraint	Aripiprazole (3), Ambrisentan(4)

Table 9-9 Parameters used by REACTS

The LCS pruning component in REACTS used the following parameters shown in Table 9-10 below which are unchanged from the LCS parameters tested previously:

Descriptor	**Minimum Value**	**Maximum Value**	**Number of partitions**
Molecular weight	140	520	20
LogP	-0.6	5.6	20

Table 9-10 REACTS LCS pruning partitions

Only a single run was carried out for each rediscovery problem. This was to demonstrate the real-world use case of running the algorithm once and collecting the results.

9.6.2 Results

9.6.2.1 Maximum score

From Table 9-11 below, the REACTS algorithm successfully solved both problems. This is due to the pruning from the RV prune, the improved objective scoring occurring from the ML simulation, and the use of LCS pruning to narrow down the number of compounds generated during the search. Moreover, the increased number of iterations allowed the search enough iterations to traverse the space successfully. It should be noted that both the RV prune value and the LCS values were not parameterised further. Further improvements to further decrease which iteration the maximum score was discovered could be made by tuning the RV prune per problem and better partitioning of the LCS filter.

Search	Problem	Maximum Score	Iteration Discovered
REACTS	Aripiprazole	1.0	60
REACTS	Ambrisentan	1.0	150

Table 9-11 Maximum score and iteration the maximum score was discovered by REACTS

9.7 Conclusions

In this chapter, several new modifications to the RVMCTS were introduced. These modifications were introduced directly to improve the overall performance of the search and to overcome the limitations identified in Chapters 7 and 8. These modifications were Max-SAUCT, a simulated annealing approach to exploration as part of the UCT equation. PUCT, predictor UCT relying upon the probabilities from a CB model. Pruning via PhysChem and LCS filtering. Finally, pruning via the CB learned model. The modifications herein, promisingly demonstrate that the approach of tree searching utilising a modified tree search approach can solve rediscovery problems. Several key results were identified: The UCT equation can be effectively replaced with an automated or adaptive exploration and exploitation. Physical chemical descriptors and LCS filtering can help reduce the total number of compounds that are being generated and exist in the tree. RV pruning is essential for managing the combinatorial explosion of products that are being generated. Finally, utilisation of pre-computed RV enumerations can be leveraged to improve the overall quality of the search. However, care needs to be taken on how these datasets are constructed and used with the CB approach.

A fusion of SAUCT and PUCT was proposed controlling the amount of exploration with a T value which slowly decreases over successive iterations lowering exploration dynamically.

Finally, the best performing modifications were combined along with the ML simulation approach in Chapter 8. This combined approach known as REACTS (REACtion Tree Search) successfully solved both aripiprazole and ambrisentan problems.

In the next chapter REACTS will be used in several design scenarios more indicative of real-world use applications.

Chapter 10 Testing REACTS

10.1 Introduction

In Chapter 9, a series of modifications were introduced to the search algorithm. The modifications were then combined to form REACTS, which was used to solve the two rediscovery problems. However, testing of REACTS was performed only on rediscovery problems using a pre-seeded database, reagent pool, and a SM which is guaranteed to be transformable into the final goal compound. This chapter will therefore focus on testing REACTS on a different set of de novo design problems. In particular, the problems in this chapter are based on searching for molecules, where the best molecule is not known explicitly.

10.1.1 Testing de novo design algorithms

As highlighted in Chapter 3 and summarised here, there is no one unifying approach to testing a de novo design algorithm. Recently, the focus of testing de novo design algorithms has been to use de novo design benchmarks (Brown et al., 2019). These benchmarks have been designed with the goal of making algorithms comparable by providing a set of predefined problems. These benchmarks have become popular as a turnkey solution to testing de novo design algorithms as they standardise the scoring functions and standardise the dataset for model training. Additionally, these single objectives are usually combinations of molecular descriptors such as QED. QED is the quantitative estimation of drug-likeness (Bickerton et al., 2012) and is a measure of "Chemical beauty" based on a series of descriptors. Single objective scoring functions are useful as diagnostic measures to demonstrate that a de novo design algorithm can find molecules with high scores.

In reaction-based de novo design, the most convincing demonstration of de novo design success is via the prospective design of compounds (Button et al., 2019; Hartenfeller et al., 2012). This approach, however, requires significant resources. Therefore, retrospective validation is often performed (Ghiandoni, 2019; Gottipati et al., 2020; Horwood & Noutahi, 2020). Retrospective validation is where known drug molecules are set as goals, and the algorithm is tested to see if it can generate the goal compound from a given starting point.

In this chapter, a series of design scenarios are posed to test the practical applicability of the REACTS algorithm. The goal is to demonstrate that the tree search algorithm can propose compounds that match a predefined property profile.

10.1.2 Scoring function definition

To test REACTS, seven different problems have been explored. The first three problems are single objective problems that form part of the Brown et al. benchmark. Problems 4-7 are a set of four separate hamming distance minimisation problems to known drugs which were part of a prospective design study by (Button et al., 2019). The goal of each of problems 1-3 is to maximise the objective, and the goal of problems 4-7 is to minimise the distance, respectively.

Problem one is the Central Nervous System MultiParameter Optimisation (CNS MPO) objective as defined by (Brown et al., 2019) and is an adaption of a more complex CNS MPO developed by Pfizer (Wager et al., 2010). The CNS MPO score is based on the physical-chemical characteristics of a molecule. The physical-chemical characteristics are topological polar surface area (40 to 90), number of hydrogen bond donors (0 to 2), LogP ($\leq$5.0), and molecular weight (<360). Each of the objectives is transformed using a separate desirability function. For example, if the LogP of a molecule is less than or equal to the value of 5.0, a score of 1.0 is given; otherwise, the score decays based on a Gaussian modifier. This transformation approach is applied separately for each physical-chemical descriptor. Finally, the CNS MPO score is formed by scalarisation of the descriptors by combining via the arithmetic means of the transformed scores. The CNS MPO score is implemented directly from (Brown et al., 2019) publicly available source code (*BenevolentAI/GuacaMol: Benchmarks for Generative Chemistry*, 2021).

Problem two is a quantitative estimate of drug-likeness (QED) score maximisation. The objective is to maximise the QED score as defined by (Bickerton et al., 2012). QED is a desirability function created to quantify as a continuous value the drug-likeness of a molecule. Molecules with a higher QED score are seen as more drug-like and, therefore, desirable. The QED score encompasses several physical-chemical characteristics such as mass weight, LogP, number of hydrogen bond donors, number of hydrogen bond acceptors, number of rotatable bonds, polar surface area, number of aromatic rings, and number of undesirable substructures. These characteristics are then transformed using several weighting functions. Finally, the weighted scores are then transformed using an exponent function to give a final value. The QED score used is based on an open-source implementation (*RDKit*, 2021).

Problem three is scaffold hopping. Scaffold hopping attempts to maximise the pharmacophoric similarity to a predefined molecule while avoiding a predefined scaffold and including a defined set of decorators/substituents. The scaffold to avoid is shown by the reference molecule, and decorators for the scaffold hopping problem in Figure 10-1. The CNS MPO score is implemented directly from (Brown et al., 2019) publicly available source code (*BenevolentAI/GuacaMol: Benchmarks for Generative Chemistry*, 2021).

Problems 4-7 are a set de novo design problems which were explored by (Button et al., 2019) as part of a wider prospective design study of the reaction-based de novo design software DINGOS. The goal of the prospective study was to generate compounds using DINGOS, which minimised the hamming distance to a known drug molecule. Four molecules were explored as part of the study (Button et al., 2019) and are alectinib, cariprazine, osimertinib and pimavanserin (shown in Figure 10-2). The designed compounds proposed were assessed via the hamming distance metric using the 166 MACCS Keys fingerprint. For this work, the same four compounds are used as targets and will be assessed using the same criteria with the results compared to DINGOS. The compounds, however, will not be synthesised.

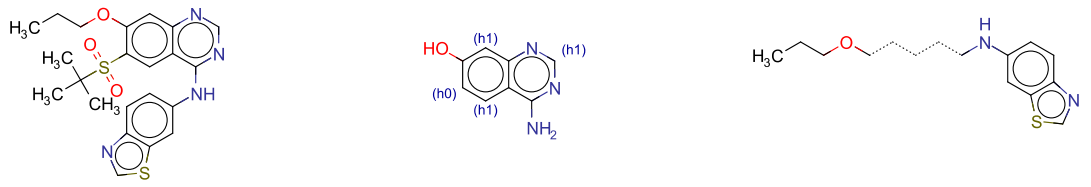


Figure 10-1 details the scaffold hopping problem used as part of the testing of REACTS. The reference molecule is the starting molecule to "hop" from by replacing the central scaffold and maintaining the decorators. The dashed line in the middle of the decorators depicts that a molecule will exist between the ether and the heterocycle decorator.

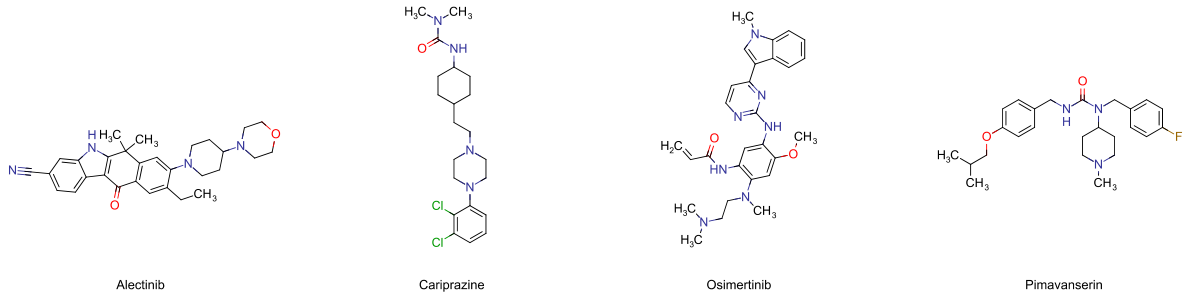


Figure 10-2 The four compounds which make up the hamming distance minimisation de novo design problems 4-7.

A summary of the different objectives are shown in Table 10-1 below:

Problem Number	Objective	Description	Reference to the scoring function	Scoring function minimum and maximum
1	CNS MPO	Maximisation of CNS MPO	(Brown et al., 2019; Wager et al., 2010)	[0-1]
2	QED maximisation	Maximisation of Quantitative Estimation of Drug-likeness (QED)	(Bickerton et al., 2012; Brown et al., 2019)	[0-1]
3	Scaffold Hopping	Maximisation of similarity to a goal compound while excluding the SMARTS pattern of a specific scaffold	(Brown et al., 2019)	[0-1]
4	Alectinib	Minimise hamming distance to Alectinib	(Button et al., 2019)	[0-1]
5	Cariprazine	Minimise hamming distance to Cariprazine	(Button et al., 2019)	[0-1]
6	Osimertinib	Minimise hamming distance to Osimertinib	(Button et al., 2019)	[0-1]
7	Pimavanserin	Minimise hamming distance to Pimavanserin	(Button et al., 2019)	[0-1]

Table 10-1 Scoring function definitions used to test REACTS

10.2 Methods

This section details the methodology used to test REACTS. The methodology is split into two parts: 1) scoring function construction, 2) description of the processes of building the dataset used to train the CB models and the simulation models used during the testing of REACTS.

10.2.1 Scoring function construction

For problems 1-3, the scoring functions are implemented using the publicly available code provided with the GuacaMol benchmark. Problems 4-7 rely on the minimisation of the hamming distance to a known drug molecule. In REACTS a minimisation problem needs to be transformed into a maximisation problem. To do this the hamming distance is transformed using equation (19). It is important note that the transformed HM bounds the problem to [0.5-1].

$$Transformed_{HM} = \frac{1}{1 + HM_{MACCS}} \quad (19)$$

Where HM_{MACCS} is the hamming distance to the MACCS fingerprint of the reference compound. This transformed Hamming distance acts as a proxy scoring function compared to raw minimisation of the hamming distance.

10.2.2 Building the CB models and the simulation models

For each of the problems outlined in Table 10-1, a simulation model and a CB model need to be created. Each model requires datasets of SMs and reagents, RVs and an appropriate scoring function. The models were created according to the workflow shown in Figure 10-3. 1) A reagent pool was selected. 2) A set of SMs was selected. 3) A set of RVs was selected and applied to the SMs and reagents to generate the SM-RV-product triplets. 4) The dataset was cleaned. 5) The products were scored. 6) The CB model was trained. 7) The simulation model was trained.

The reagents, SMs and RVs were the same for all problems so that steps one to four were performed once. Then, the product scoring and model building steps were repeated for each problem.

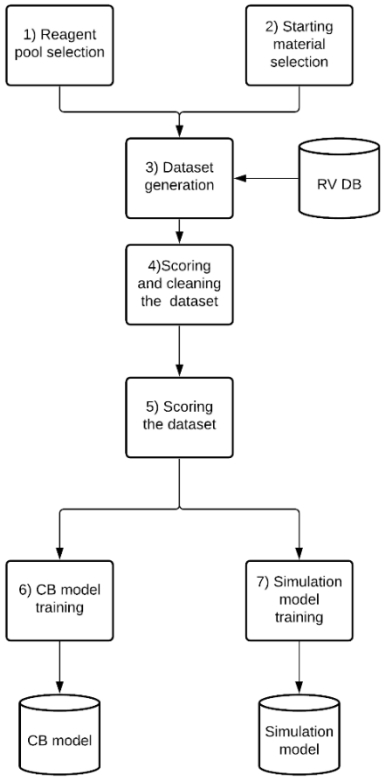


Figure 10-3 Workflow used to create the CB models and the simulation models. Notably, the 1) Reagent pool selection, SM selection, dataset generation and cleaning of the dataset were the same for all problems.

10.2.3 Reagent Pool selection

The reagent pool for testing REACTS was chosen based on filtering of the GuacaMol training set. The GuacaMol training set comprises 1.27M molecules retrieved from ChEMBL24. The GuacaMol training set was chosen as the molecules are already cleaned and processed. Notably, (Brown et al.,

2019) also removed molecules from the training set with high similarity to molecules that are part of their rediscovery benchmark. One of their rediscovery benchmark compounds is osimertinib. Osimertinib is problem six in this study. The decision was taken to use the GuacaMol dataset as osimertinib is only a single example used in this study. Moreover, using the GuacaMol benchmark facilitates better comparison to other de novo design approaches.

The GuacaMol dataset was filtered sequentially over four steps. Step 1) Molecules were removed if they violated the Astex Ro3 filters (Congreve et al., 2003). The Ro3 filters retain any molecules based on AMW (≤300), LogP (≤3), TPSA (≤60), number of hydrogen bond donors (≤3), number of hydrogen bond acceptors (≤3). Step2) Molecules were removed if they violated any of the Walters rd_filters. Walters rd_filters are a curated collection of functional group SMARTS patterns derived from several public structural alerts (Walters, 2018). Step 3) Duplicate molecules were removed based on their canonical smiles. Step 4) The top 2500 molecules were selected based on the RankSynth score. RankSynth was created for this work and is a consensus score of three synthetic tractability metrics. The synthetic tractability metrics are SAScore (Ertl & Schuffenhauer, 2009), SCScore (Coley et al., 2018), and RAScore (Thakkar et al., 2021). Each molecule was scored and ranked (rank one is the best, rank n is the worst) using each of the tractability metrics. The ranks were then summed to form the RankSynth score. The top 2500 reagents based on their RankSynth scores were selected. The molecules with the best and worst RankSynth scores computed during step four are shown in Figure 10-4 below:

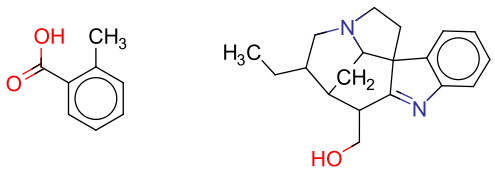


Figure 10-4 Best(left) and worst(right) RankSynth molecules before final step filtering

A visual depiction of the process of generating the reagent pool is detailed in Figure 10-5 below:

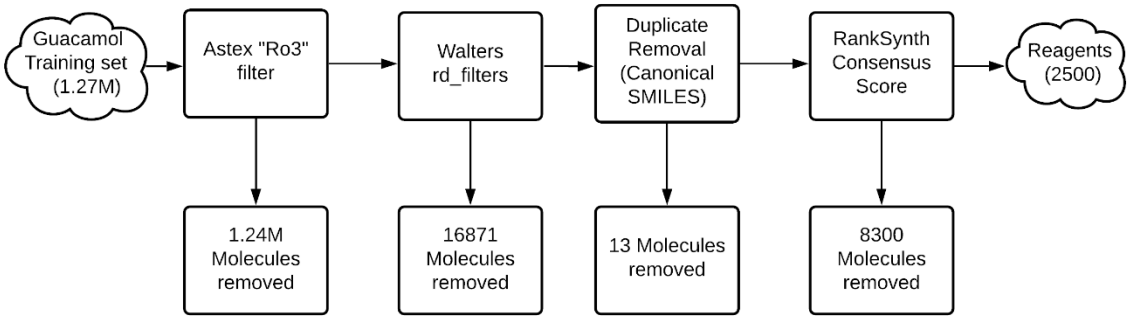


Figure 10-5 A Workflow to filter the GuacaMol testing set down to 2500 reagents used for the dataset generation step. Step one filtered all compounds based on the Astex Ro3 filter. Step two removed compounds that contained undesirable substructures based on the curated walters rd_filters. Step three removed compounds that were duplicates based on their canonical SMILES. In Step four, the remaining molecules were scored with RankSynth, and the top 2500 compounds were retained to form the reagent pool.

10.2.4 Starting material selection

The SMs (250k) were selected by filtering the GuacaMol training set (1.27M) using a sequential four-step workflow. Step 1) Molecules were removed if their mass weight exceeded 500 da. Step 2) The molecules were filtered if the LogP exceeded 5.0. Step 3) Duplicate molecules were removed based on their canonical SMILES. Step 4) the top 250k molecules were selected based on their RankSynth score (as described previously) (shown in Figure 10-6).

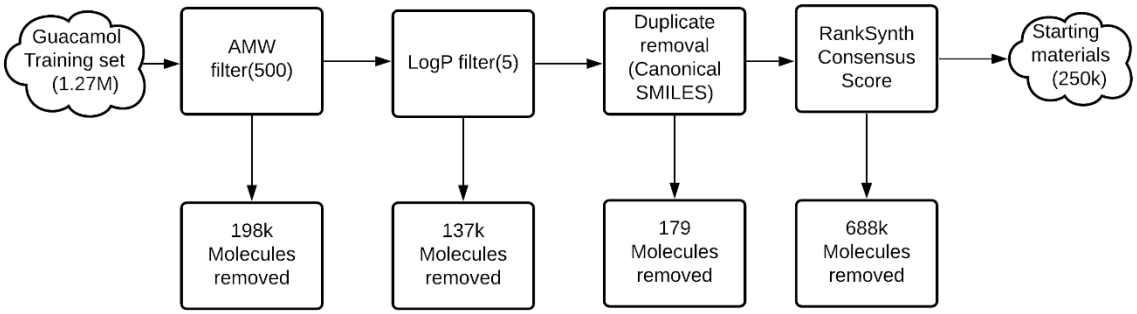


Figure 10-6 filtering of the GuacaMol training data to prepare SMs for expansion with the filtered reagent pool. Step one removed compounds with mass weight greater than 500. Step two removed compounds with a logP greater than five. Step three Duplicate molecules were then removed based on their canonical SMILES. Step four Molecules were ranked via RankSynth, and the top 250k molecules were selected.

10.2.5 Dataset generation

The expansion of the 250k SMs with the 2.5k reagents and the JMedChem 4.5k RVs took 3 hours and 15 minutes on the Sheffield HPC. The total number of generated products was 70.2M.

10.2.6 Cleaning the dataset

The generated dataset was then cleaned. Cleaning of the dataset followed a two-step process. First, all invalid structures were removed, after which 6.18M duplicate (SM- RV-product) triplets were removed. Thus, the final dataset consisted of 63.2M unique SM- RV-product triplets. The filtering workflow is shown in Figure 10-7.

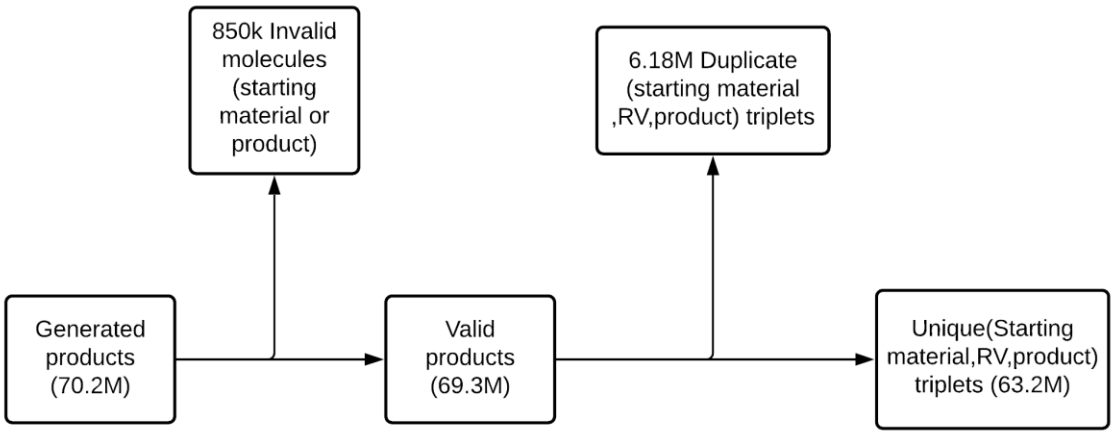


Figure 10-7 Step one Invalid molecules were removed; these were molecules which the RDKit could not read. Step two duplicate compounds were removed based on SM, RV , product triplets. The molecules are then scored. The scored molecules are then separated into a CB training dataset (3.09M) and a simulation model training dataset(243k). In total, four CB model datasets were generated, and four ML mean simulation datasets were generated (two per scoring function).

10.2.7 Scoring the dataset

The cleaned products were then scored and transformed into two separate datasets (shown in Figure 10-8). One dataset was used to train the CB model (3.09M examples), and one dataset was used to train the simulation model (243k). The 3.09M examples are generated by the groupby process which was used to collect all unique (SM-RV) pairs. The 243k are unique SMs with corresponding mean scores of all product scores (7k SMs were removed during the cleaning process shown in section 10.2.6). This process was repeated for each scoring function.

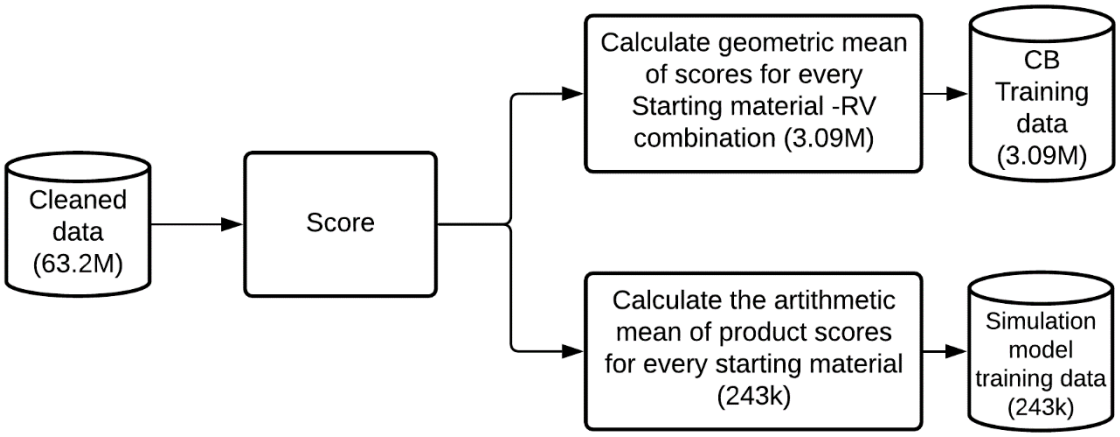


Figure 10-8 Scoring of the cleaned dataset.

10.2.8 Training the simulation model

After collection and cleaning of the data, the simulation models were created. The process was the same as outlined in Chapter 8, and one model was trained for each scoring function. In addition, a 90:10 cross-validation (10-fold CV) was performed to determine the most appropriate bit length and radius for the input Morgan fingerprint. The complete list of CV model parameters and corresponding standard deviations are shown in appendix H. Table 10-2 below shows the CV model performance of the best performing model along with the final trained model performance tested on an internally sampled holdout set. It is important to note that in the scaffold hopping ML model, the best Morgan fingerprint was 16384 bits with radius=3 based on the CV, whereas, for the other problems, the best input fingerprint was 16384 bits with a radius=2. Finally, in the case of the scaffold hopping simulation model, the R^2 score was slightly improved on the holdout data, as shown in Table 10-2:

Problem	**Morgan Fingerprint**	**CV model R^2 score**	**Final model R^2 performance**
CNS	16384Bits, Radius=2	0.726 ± 0.003	0.696
QED	16384Bits, Radius=2	0.752 ± 0.002	0.720
Scaffold Hopping	16384Bits, Radius=3	0.63 ± 0.007	0.690
Alectinib	4096Bits, Radius=2	0.777±0.004	0.769
Cariprazine	4096Bits, Radius=2	0.823±0.003	0.821
Osimertinib	8192Bits, Radius=2	0.761±0.004	0.761
Pimavanserin	4096Bits, Radius=2	0.819±0.003	0.813

Table 10-2 Cross validation and corresponding final model R^2 performance for the ML mean simulation used during REACTS

Each 10-fold cross validation took approximately two minutes to complete. In total for each problem the total training time was 20 minutes (2 minutes for each 10-fold CV per representation).

10.2.9 CB model training

After scoring the cleaned dataset, the CB models for each of the problems were created. The goal of the training is to build a CB model which selects better performing RVs based on the geometric mean of their product scores. The training of each CB model occurred for 1M training steps and used the same process as outlined in Chapter 9, this time using the dataset generated in section 10.2.7.

Table 10-3 outlines the CB models average reward. If the CB models average reward is lower than the data mean, the model is not successful in choosing RVs better than random. Promisingly, in every case, the CB model outperforms the data mean. In the Scaffold hopping and transformed hamming distance minimisation problems, the CB model had a relatively small improvement over

the data mean (<=0.05). This low figure would suggest that these CB models struggled to identify a relationship between SM and RV to geometric mean score during training.

Problem	CB Model	Data mean
CNS MPO	0.866	0.826
QED	0.595	0.501
Scaffold Hopping	0.391	0.389
Alectinib	0.787	0.783
Cariprazine	0.793	0.788
Osimertinib	0.785	0.780
Pimavanserin	0.785	0.781

Table 10-3 Final CB model performance used by each of the REACTS searches.

For each of the CB models the training took approximately, 3hrs per model to training for 1M training steps.

10.2.10 Starting material selection

Reaction-based de novo design needs a SM. Acetic acid (Figure 10-9) was chosen as the SM because it contains a simple functional group that can be transformed in several ways or bound to larger reagents. Moreover, choosing a consistent starting point prevents benchmarking bias from being introduced into the search whereby a SM is chosen that is already close to the goal objective. Finally, it allows comparison between REACTS searches using different scoring functions to see if different regions of chemical space are being traversed.

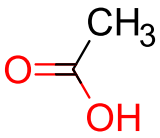


Figure 10-9 Depiction of Acetic acid

10.2.11 REACTS parameter selection

The following REACTS parameters were used. In this chapter, the number of iterations was increased to 10000 to provide REACTS enough computational time to explore and exploit the search space. The RV database was the JMedChem 4.5k with no seeded reactions. The reagent pool was the GuacaMol cleaned 2500 reagent set. The depth constraint was set to ten steps as synthetic schemes longer than ten steps would be classed as undesirable due to poor yield.

Parameter	Value
Iterations	10000
Exploration constant (C_p)	0.7071
RV database	JMedChem 4.5k
Reagents	GuacaMol cleaned 2500
RV prune	k=10
Depth constraint	10

Table 10-4 Parameters used during REACTS searches.

The LCS pruning component in REACTS used the following parameters shown in Table 10-5 below. In addition, the number of partitions was increased relative to the previous chapter to promote molecular diversity during the search.

Descriptor	Minimum Value	Maximum Value	Number of partitions
Molecular weight	140	520	40
LogP	-0.4	5.6	40

Table 10-5 REACTS LCS partitions used during the searches.

In this work, a single replicate was run to reflect the likely real-world scenario of running the algorithm once without replicates.

10.3 Results

The results are presented for each problem in turn. In the case of the transformed hamming distance problems these are presented together. Where possible, a comparison to established methods is provided.

10.3.1 CNS MPO

The performance of REACTS on the CNS MPO problem is shown in Table 10-6 below. Firstly, the search generated 214 molecules with a maximum score of 1.0. The performance of REACTS is comparable to the performance of alternative DNN approaches described in the GuacaMol

benchmark (Brown et al., 2019), which also scored a maximum of 1.0. To understand the search performance, it is helpful to inspect and contrast the best and worst molecules generated. Figure 10-10 shows example molecules with the best and worst scores found in the REACTS tree. There is a stark difference in the size of the best performing molecule compared to the worst-performing molecule. Figure 10-11 shows the distribution of CNS MPO scores. The distribution is close to the maximum value of one. However, the tail of the distribution is relatively short, suggesting that the search has focussed on molecules with relatively high scores.

Max	Min	Mean	Std.dev	Variance	Skewness	Kurtosis	Maximum solutions	Total molecules generated
1.0	0.418	0.882	0.103	0.011	-1.124	1.234	214	3098

Table 10-6 Summary statistics for the CNS MPO REACTS tree

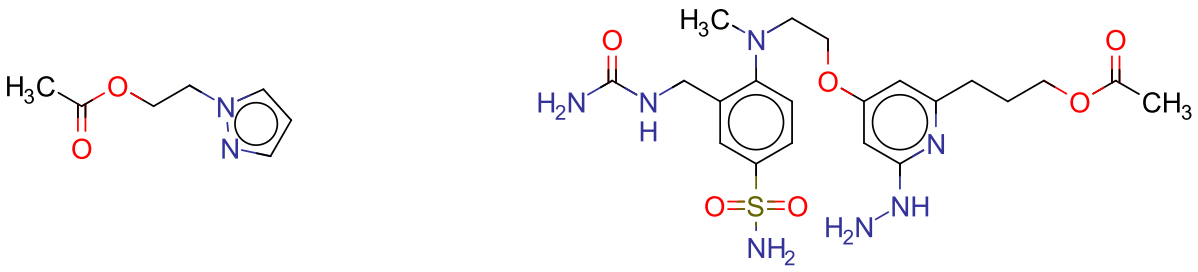


Figure 10-10 Best and worst molecules found in the tree. The molecule on the left has a score of 1.0, whereas the molecule on the right has a score of 0.418

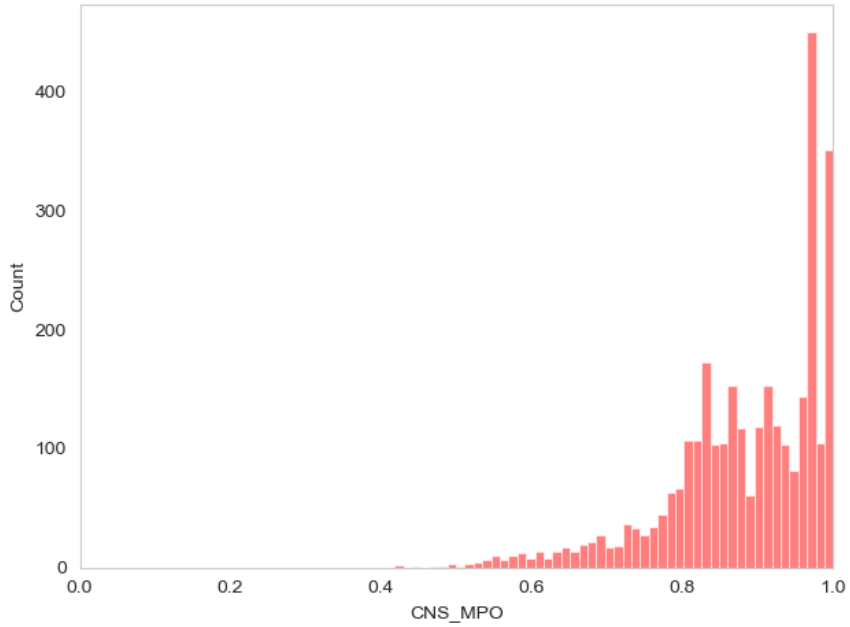


Figure 10-11 Distribution of CNS MPO score

Of the 214 solutions, a further breakdown can be obtained by observing where in the tree they were generated. Table 10-7 shows the summary statistics for the tree depth of the molecules with a score of 1.0. Remarkably, the solutions were found at a wide range of depths. The shallowest solution occurred after a single RV was applied. The deepest solution occurred at depth ten after ten RVs were applied. Moreover, the mean depth is relatively shallow (~4 steps). This is promising as it suggests that even though REACTS was configured to grow to a greater maximum depth, the search focussed on shallower regions where the highest-scoring molecules were found.

Minimum Depth	Maximum Depth	Mean Depth	Standard deviation
1	10	3.860	1.64

Table 10-7 Summary statistics of the 214 solutions found by REACTS during the CNS MPO search

Figure 10-12 and Figure 10-13 show the reaction sequences for solutions found at depths one and ten, respectively. In the depth one example, the molecule found resulted from a relatively small change using the reagent shown. Remarkably the solution found at depth ten starts from an ester formation and goes through several steps until finally forming the amine. This longer path demonstrates REACTS ability to traverse much deeper search trees than have previously been tested. Finally, Figure 10-14 shows the tree generated by REACTS. The tree, in contrast to the tree generated in the previous chapters, is much deeper. Interestingly, the scores of the generated molecules decrease as the tree gets deeper. This is to be expected due to the increase in mass resulting from more applications of RVs and the search moving away from desirable regions of chemical space that match the CNS MPO score.

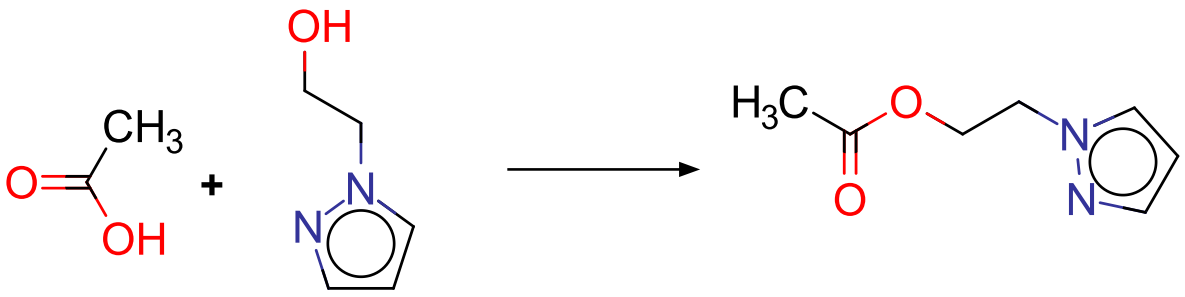


Figure 10-12 Depth one solution found for the CNS MPO problem by REACTS

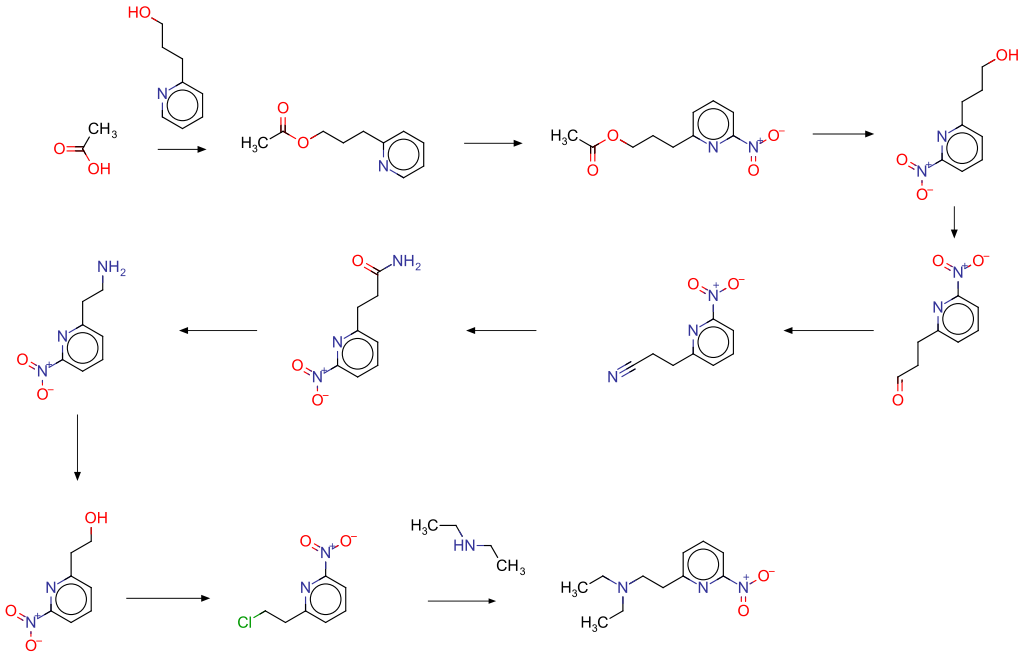


Figure 10-13 Depth ten solution found for the CNS MPO problem by REACTS

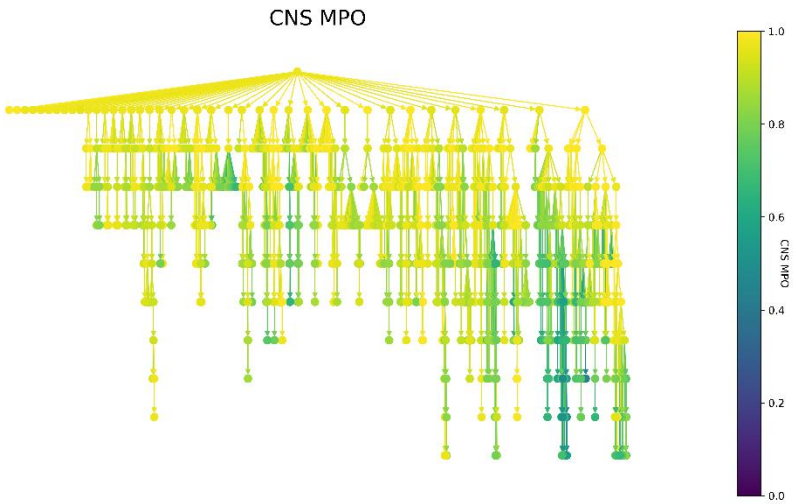


Figure 10-14 CNS MPO Tree generated by REACTS

An essential consideration in reaction-based de novo design is the synthetic tractability of the molecules that are being generated. Three scores are used to indicate the synthetic tractability of the generated molecules. These are: SAScore (Ertl & Schuffenhauer, 2009), SCScore (Coley et al., 2018), and RAScore (Thakkar et al., 2021).These scores were used in sections (10.2.3 and 10.2.4) as part of RankSynth. Figure 10-15 shows the summary statistics for each of the computed synthetic assessment scores. In all cases, each approach suggests that the molecules generated in the tree are synthetically tractable. In particular, the RAScore, a measure of retrosynthetic feasibility, suggested

that only 104 out of the 3096 molecules would not be retrosynthetically solvable. Finally, the molecules generated by REACTS were searched using an exact match to the ZINC20-In-Stock library. 71 Compounds generated by REACTS were directly purchasable from ZINC20, which shows that REACTS can generate synthetically tractable molecules. Since this is a small proportion of the molecules generated, this highlights that the majority of the compounds that are being generated are likely to be novel molecules.

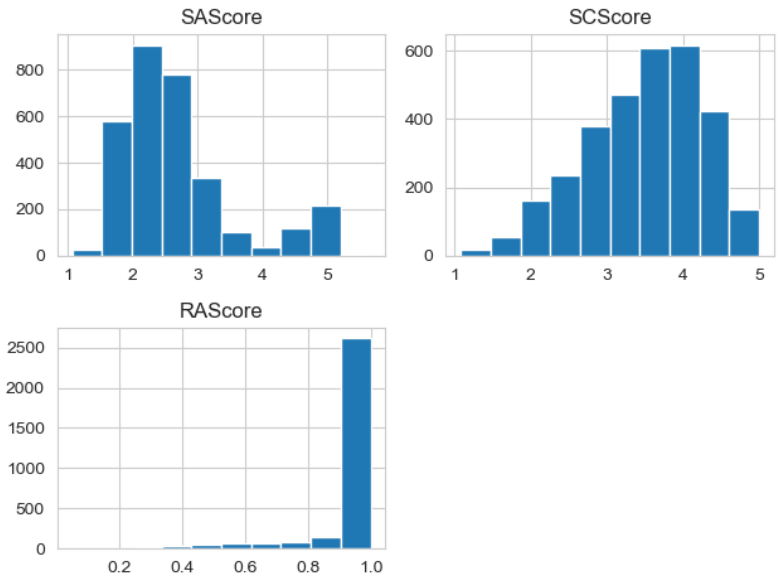


Figure 10-15 Synthetic tractability scores for the CNS MPO Tree generated by REACTS

ZINC20-in-stock (tranche)	Purchasable molecules generated by REACTS
Fragment	54
Lead	1
Drug	16

Table 10-8 Purchasable molecules generated by REACTS for the CNS MPO problem

10.3.2 QED

Table 10-9 shows the summary statistics for the QED results. The maximum QED found in the tree was 0.943. This is 0.05 QED units lower than the best molecule generated in the DNN benchmark (0.948) and three alternate RBDD DL/RL approaches (0.948) (Bradshaw et al., 2020; Gottipati et al., 2020). However, in contrast to these DL approaches, the number of reagents used by REACTS is several orders of magnitude smaller. Moreover, the number of transformations utilised by the published RBDD DL/RL approaches is several orders of magnitudes greater than used by REACTS.

Additionally, (Horwood & Noutahi, 2020), another DL/RL RBDD approach based on a smaller reagent pool (5000 molecules) derived from BRICS fragmentation of a PubChem subset resulted in a QED score of 0.947. Therefore, the value of 0.943 is a competitive QED score under the constraints of the reagent and RV pools. The worst value found in the tree was a score of 0.149. This molecule would be highly undesirable to chemists, according to the QED metric (shown in Figure 10-16). Therefore, it is interesting to see both molecules in the tree and highlights REACTS ability to traverse a wide range of chemical space in search of the best scoring molecules. The synthetic scheme proposed to the best QED compound is shown in Figure 10-17. This synthetic scheme is interesting as it mimics a classic protection, functionalisation, deprotection scheme. A protection-deprotection scheme is an approach that has previously been disfavoured due to protections generally decreasing the objective score. However, using QED to drive the search shows a steady improvement in the QED score from intermediate to intermediate, leading to REACTS discovering the path. The distribution of QED scores is shown in Figure 10-18. The distribution of QED scores shows less skew towards higher scoring molecules than was seen for the CNS MPO results, and this is likely due to it being harder to optimise the QED score relative to the CNS MPO score.

Max	Min	Mean	Std.dev	Variance	Skewness	Kurtosis	Total molecules generated
0.943	0.149	0.604	0.168	0.028	-0.309	-0.798	3376

Table 10-9 Summary statistics of the QED REACTS Tree

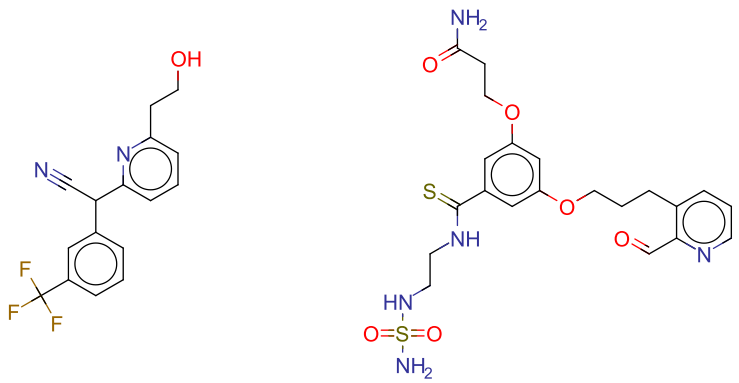


Figure 10-16 The best (left) and worst (right) QED molecules found in the REACTS Tree.

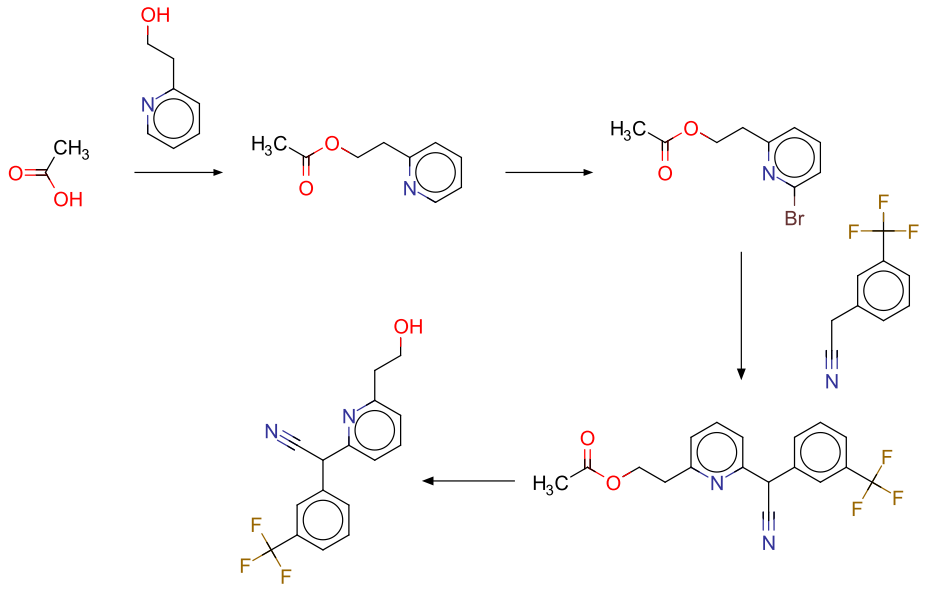


Figure 10-17 Synthetic scheme to the highest scoring QED molecule generated by the REACTS algorithm

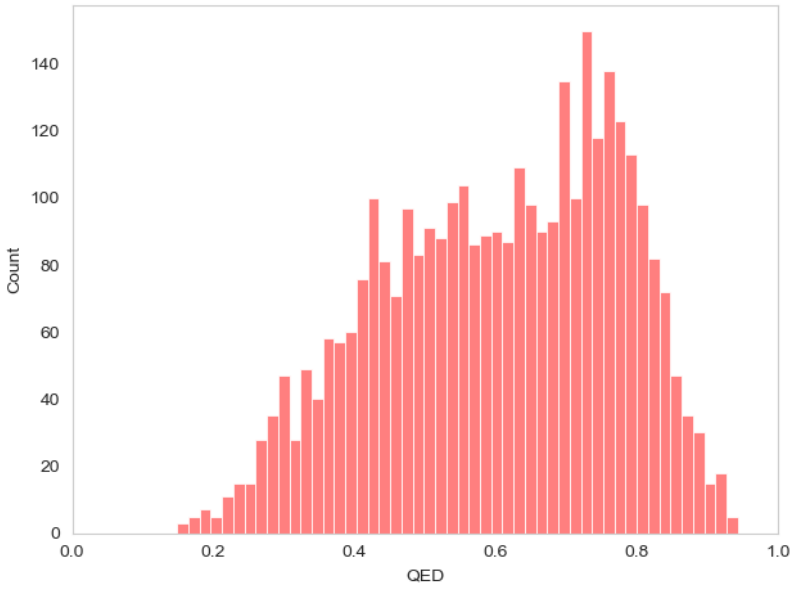
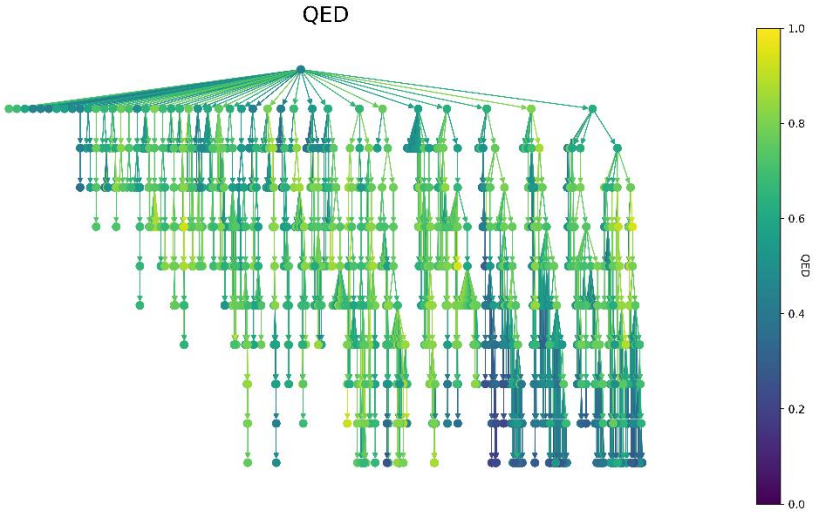


Figure 10-18 Histogram of QED distribution of molecules generated by REACTS

As described previously, the only changes between the CNS MPO search and the QED search are the corresponding simulation models, the CB models and the scoring functions from which they are derived. It would therefore be interesting to know how many molecules overlap between both searches. Table 10-10 shows the summary statistics for both the CNS MPO and QED searches and the percentage overlap compared to the QED tree search. The table highlights a key point: even though the CNS MPO and QED scoring functions are similar (they both aim to capture drug-likeness), there is minimal overlap in the total molecules generated in each search. This further confirms that the scoring function can shape the search and is aided by smarter RV prioritisation.

Total Molecules QED	**Total Molecules CNS**	**Overlap**	**Percentage Overlap (QED total)**
3376	3308	326	9.65%

Table 10-10 Overlap summary statistics between QED and CNS MPO REACTS searches



As highlighted previously, it is helpful to understand the performance of REACTS in terms of the synthetic tractability of the molecules that have been generated. Overall, the synthetic tractability is again good. 138 molecules were directly purchasable from vendors. Interestingly, the SCScore is shifted higher than for the molecules generated for the CNS MPO problem. The RAScore again demonstrated that most of the molecules are retrosynthetically tractable and therefore likely to be synthesisable. In total, only 60 molecules were predicted not to be synthesisable based on the RAScore.

ZINC20-in-stock (tranche)	**Purchasable molecules generated by REACTS for the QED search**
Fragment	105
Lead	2
Drug	31

Table 10-11 Purchasable molecules generated by REACTS for the QED problem

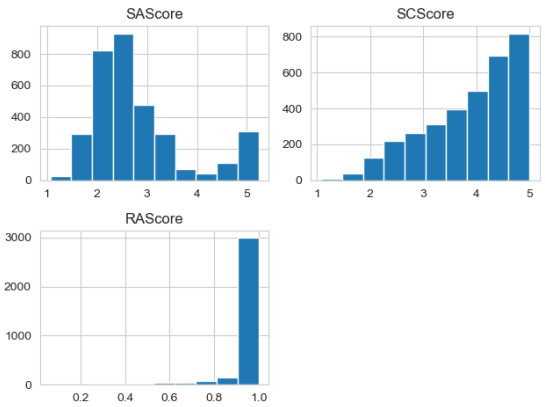


Figure 10-19 Synthetic tractability scores for the QED Tree generated by REACTS

10.3.3 Scaffold Hopping

Table 10-12 details the summary statistics for the REACTS generated tree. First, the maximum score is 0.489, which is relatively low compared to established DNN methods, which achieved a maximum of 1.0 (Brown et al., 2019). However, this is still a modest performing score given that the reagent pool did not contain the decorators expected to surround the molecule. Figure 10-20 depicts the reference structure and the top one and two performing molecules (left to right). Interestingly, REACTS has exploited the scoring function by excluding the scaffold; however, it has not found the correct scaffold or decorators. It is likely that if a SM closer to the reference compound was used together with a larger reagent pool, then REACTS would have performed better on this problem. Figure 10-21 details the path from SM to the final best scoring molecule and again shows the ability of REACTS to build up a molecule and then remove mass if it yields a better performing end structure. Figure 10-22 shows the histogram of scores. In comparison to the previous problems, the scores are firmly centred on 0.4. This is also seen in the tree visualisation shown in Figure 10-23, whereby the scores in the tree do not vary a great deal.

Max	**Min**	**Mean**	**Std.dev**	**Variance**	**Skewness**	**Kurtosis**	**Total molecules generated**
0.489	0.334	0.412	0.029	0.001	-0.324	-0.513	5845

Table 10-12 Summary statistics for the Scaffold hopping (GM) REACTS tree

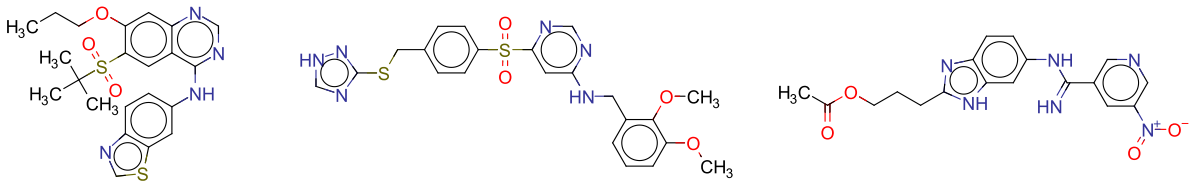


Figure 10-20 Reference molecule(left), Top 1(middle) and 2(right) compound generated by scaffold hopping REACTS.

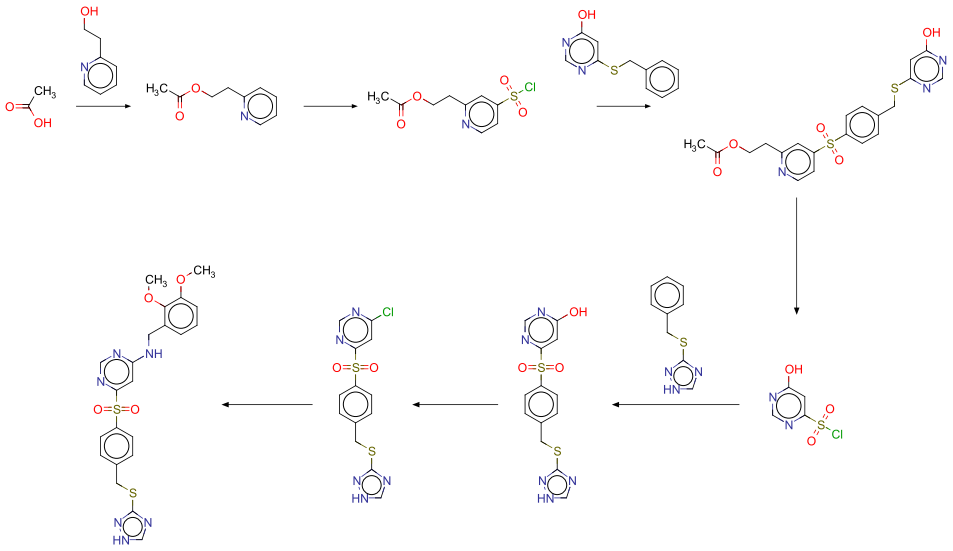


Figure 10-21 Path to the best scoring molecule proposed by REACTS for the scaffold hopping problem

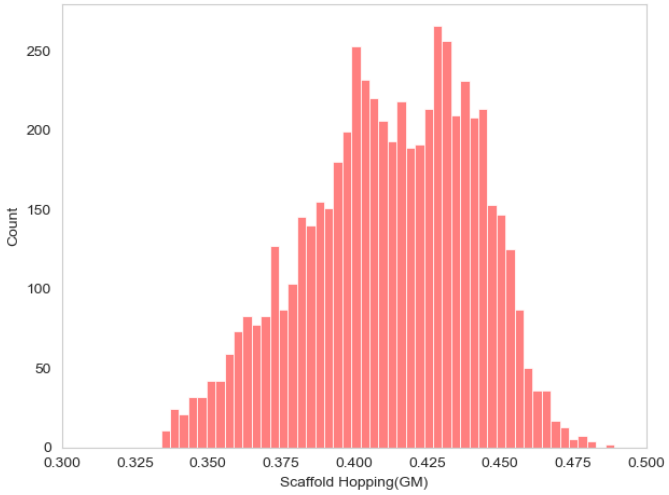


Figure 10-22 Scaffold hopping (GM) scores generated by the REACTS Tree

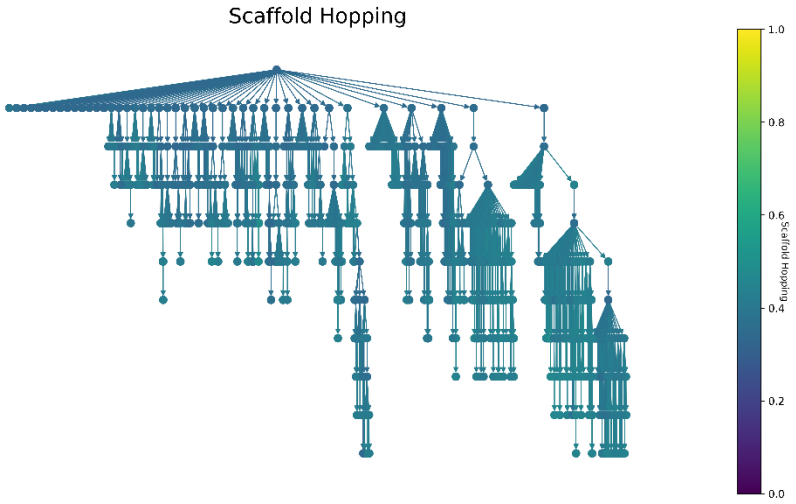


Figure 10-23 Scaffold hopping tree generated by REACTS. The consistent colouration is due to the majority of the tree existing between 0.3 and 0.4 units.

The distributions of the synthetic tractability scores are shown in Figure 10-24, where the majority of the molecules score well. Finally, again molecules that REACTS has proposed were searched in ZINC-20-In-Stock; in total, this time, 32 molecules were purchasable directly from vendors.

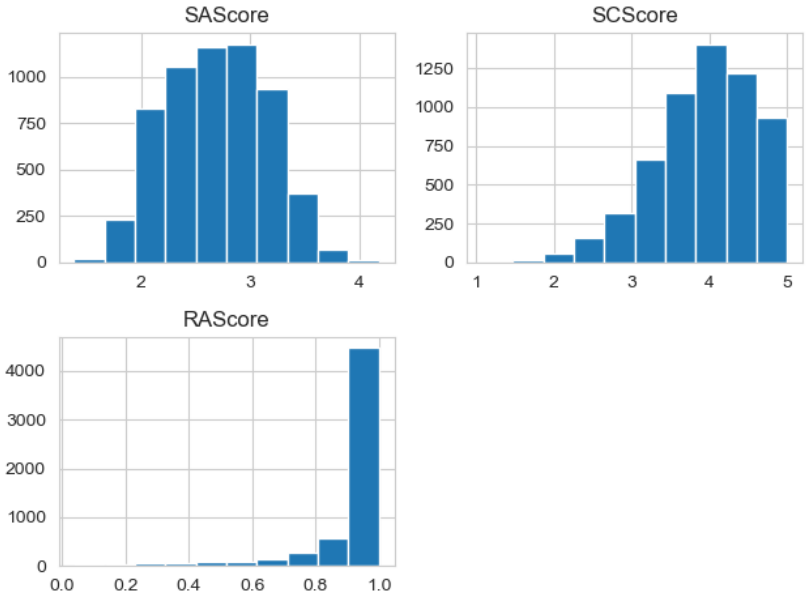


Figure 10-24 Synthetic tractability scores for the scaffold hopping tree generated by REACTS

ZINC20-in-stock (tranch)	Purchasable molecules generated by REACTS for the scaffold hopping (GM) search
Fragment	30
Lead	0
Drug	2

Table 10-13 Purchasable molecules generated by REACTS for the Scaffold hopping problem

10.3.4 Transformed Hamming distance problems

Table 10-14 shows summary statistics for the transformed Hamming distance problems. The maximum score found for each problem is relatively high. However, this is also balanced by a relatively high minimum score. Moreover, the mean of the generated trees is closer to one. This would suggest that the Hamming distance metric is a relatively simple metric to increase but a hard metric to score a value of one.

Interestingly the total number of molecules generated differs between problems. This difference in the total number of molecules generated suggests that even though the same starting point was used throughout all problems, the grown trees explored different regions of the search space. Finally, in all cases, there is a negative skewness of scores suggesting that the distributions were shifted closer to a desirable value.

Figure 10-25 shows the reference molecules (left column) and the highest scoring molecules for REACTS (centre column) and DINGOS (right column). In all cases, REACTS found compounds of comparable scores to the maximum score found by DINGOS (Button et al., 2019). This score is remarkable when considering that REACTS started with a substantially smaller starting reagent pool (2.5k for REACTS vs 214k for DINGOS), while simultaneously traversing a larger possible transformation pool (4.5k RVs Vs 64 hand crafted reaction templates). Table 10-15 shows the rank position of the best compound generated by REACTS when compared to the DINGOS scores. The rank position was within the top ten highest scoring molecules in all cases. Again, this is an additionally impressive performance considering the relatively small set of reagents and reactions. Figure 10-26 shows the proposed synthetic paths to each of the highest-scoring generated molecules for each transformed Hamming distance problem. In each case, the paths, are simple with the addition of mass as the synthetic scheme progresses. This suggests that REACTS, in this case, preferred paths which yielded coupling reactions and thus bound together larger reagents. It is interesting to note that this is different to the QED maximisation problem (see Figure 10-17), which saw REACTS prefer reaction paths which increased and then decreased mass. This would suggest that depending upon the scoring function given to REACTS, different reaction types can be

prioritised, not just reactions which increase in mass, which was a fundamental limitation of previous RV approaches. Figure 10-27 shows the Kernel Density Estimation (KDE) plot of the distribution scores generated by REACTS. This plot was created to compare the individual REACTS searches for the transformed hamming loss problem rather than the histograms used before for the QED, CNS MPO, and scaffold hopping searches. The KDE plots show that for the cariprazine problem, which generated the highest scoring molecule overall, the distribution has been shifted closer to a value of one, again confirming that REACTS can generate molecules which are closer to the design objective.

Problem	Max	Min	Mean	Std.dev	Variance	Skewness	Kurtosis	Total molecules generated
Alectinib	0.928	0.736	0.845	0.032	0.001	-0.812	0.310	3732
Cariprazine	0.954	0.726	0.863	0.037	0.001	-0.539	-0.033	5505
Osimertinib	0.928	0.714	0.840	0.034	0.001	-0.607	0.312	4240
Pimavanserin	0.928	0.752	0.852	0.028	0.001	-0.254	-0.323	5259

Table 10-14 Summary statistics of the transformed hamming loss REACTS trees

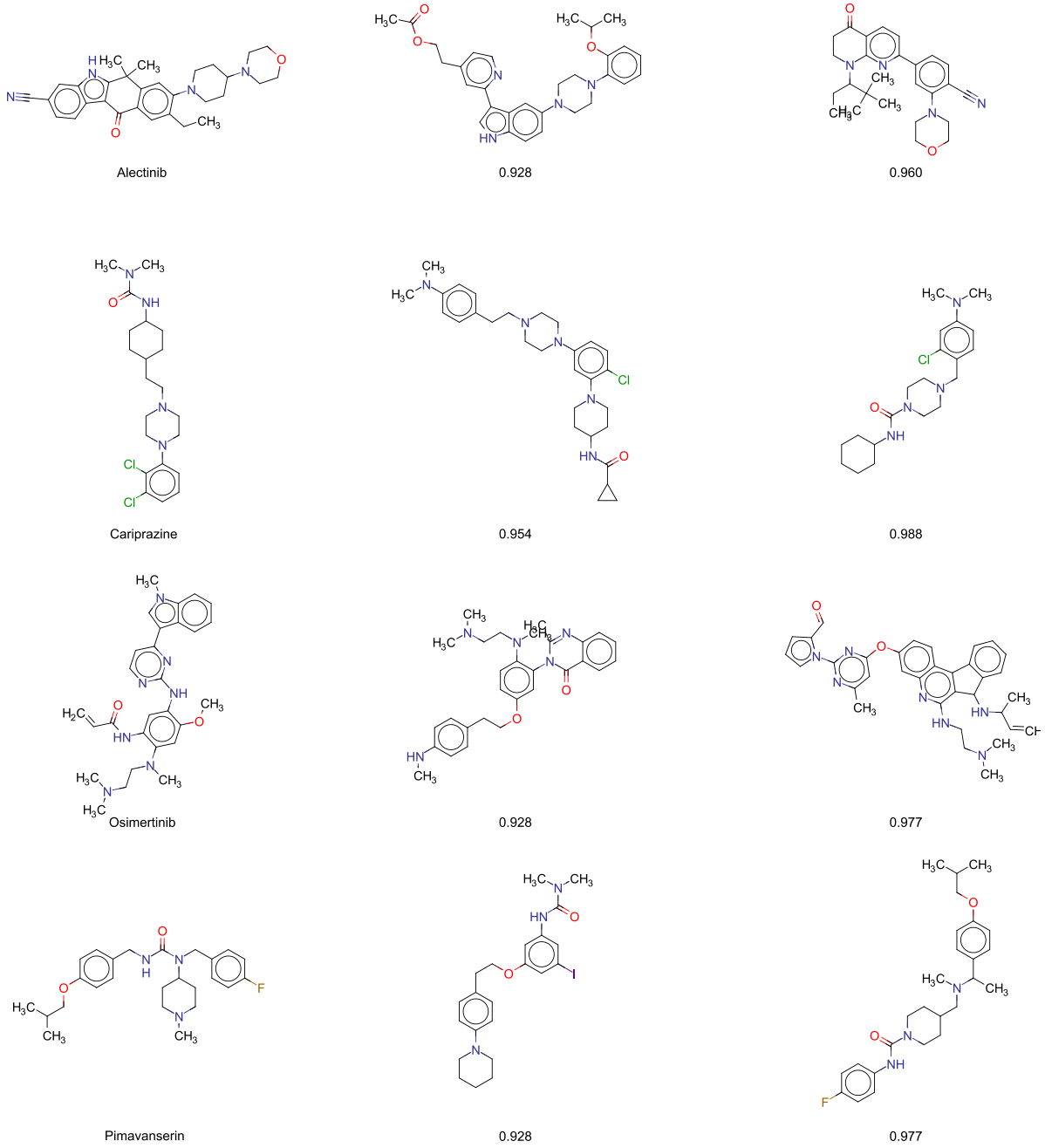
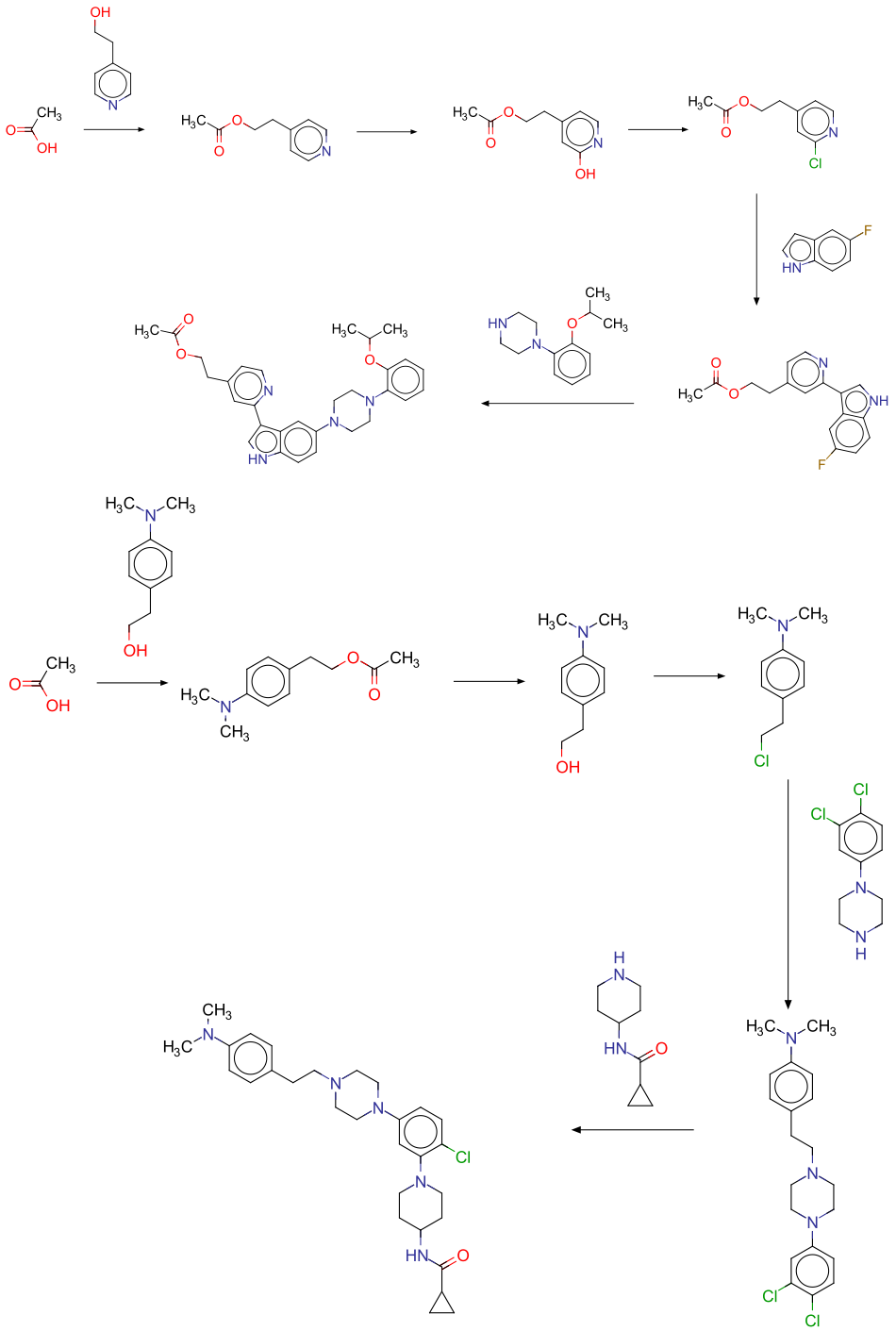


Figure 10-25 Depiction of molecules compared to their references(left). REACTS top generated molecules (centre), DINGOS top generated compound(right). The values underneath the molecules are the transformed Hamming distance metric.



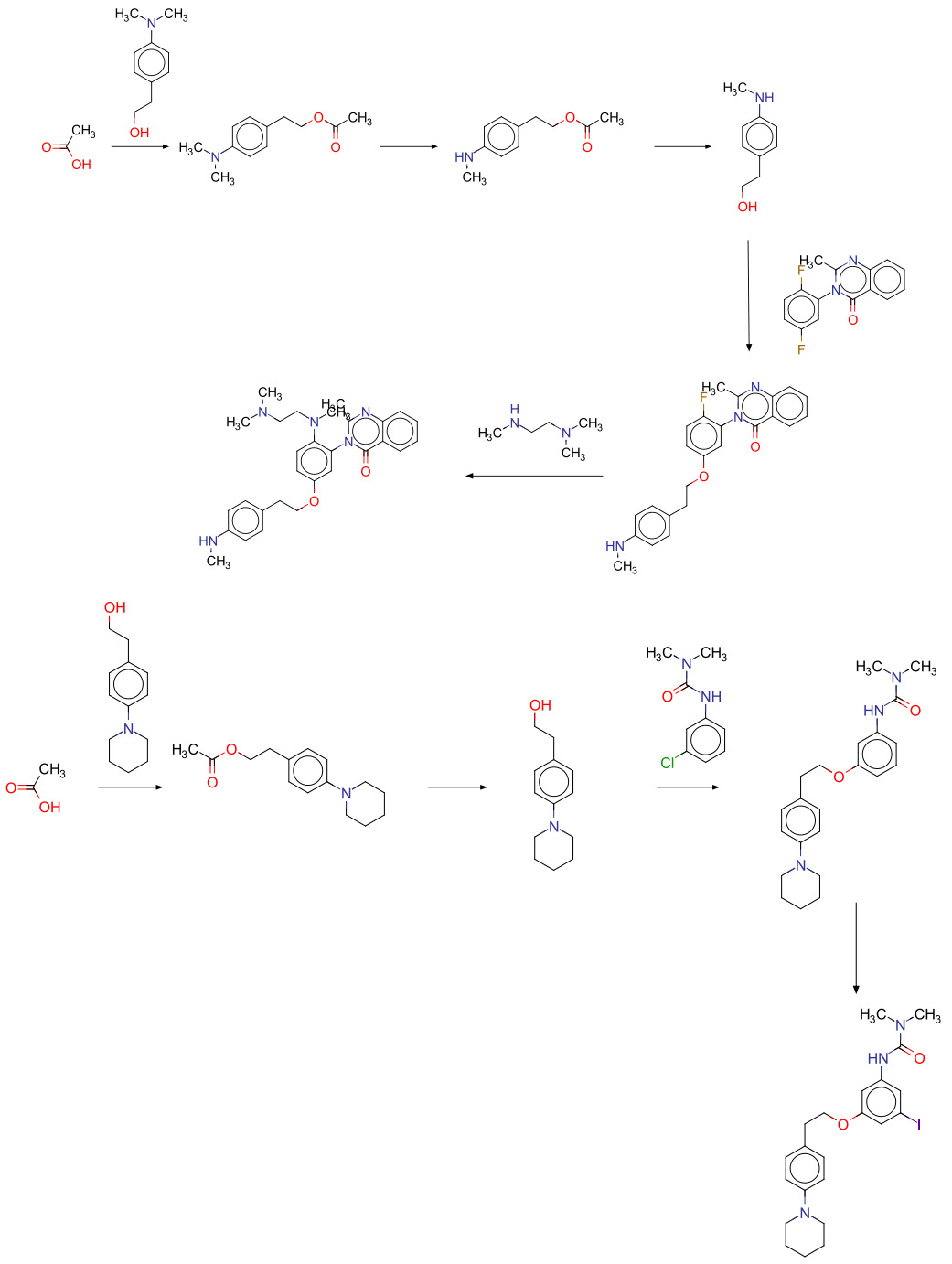


Figure 10-26 Proposed synthetic schemes generated by REACTS for each of the four hamming distance problems. 1) Alectinib, 2) cariprazine, 3) osimertinib,4) pimavanserin

Problem	Hamming distance of best scoring REACTS molecule	Best scoring REACTS molecule based on DINGOS rank	Maximum ranking available in the DINGOS dataset
Alectinib	0.078	7	23
Cariprazine	0.048	6	18
Osimertinib	0.078	7	23
Pimaversin	0.078	10	22

Table 10-15 Rank position of the best compound generated by REACTS when compared to the DINGOS rank

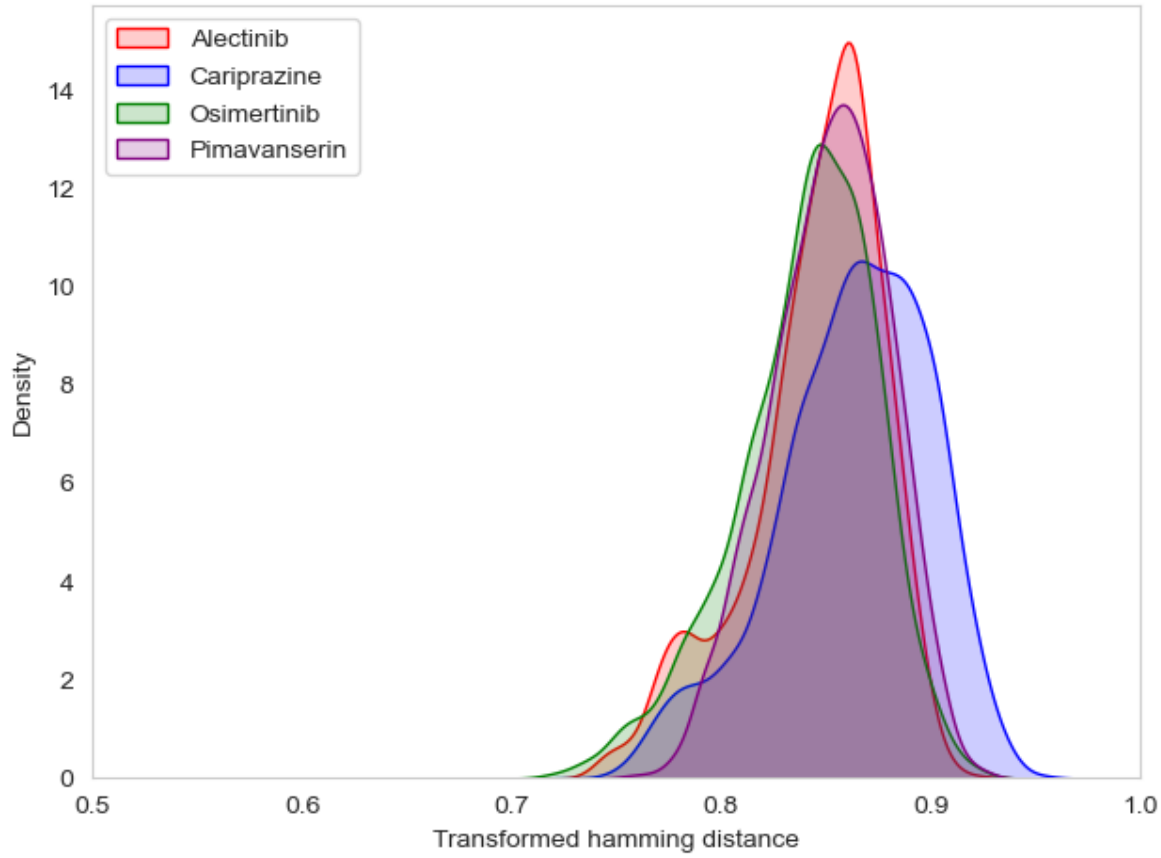
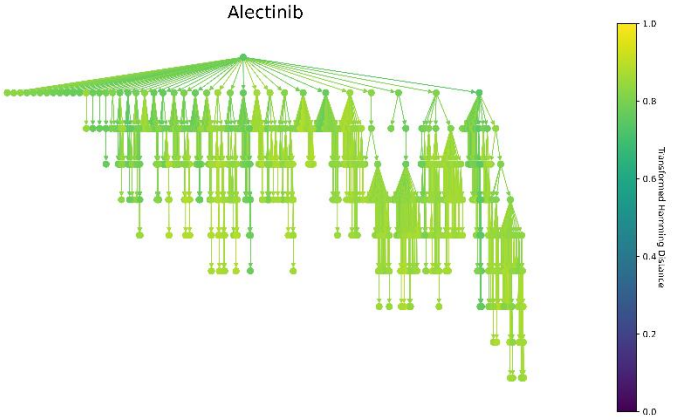
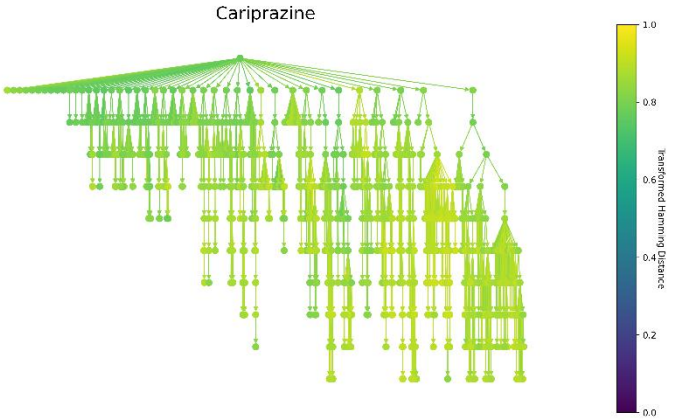


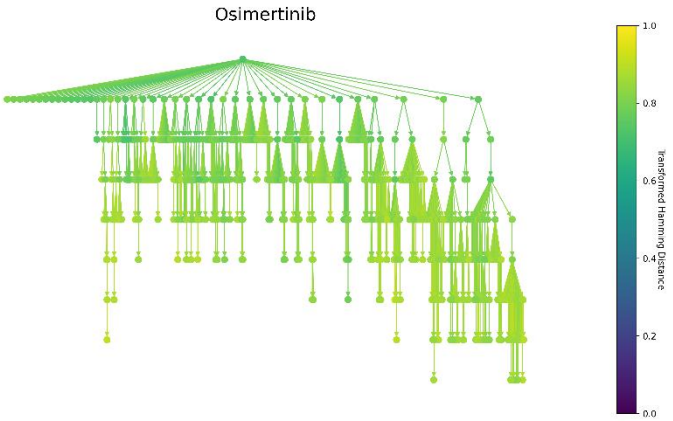
Figure 10-27 KDE plot of the distribution of scores of the generated molecules for the Transformed hamming problems

Figure 10-28 shows the trees generated by REACTS for each of the transformed Hamming distance problems. In all cases, specific paths in the tree have been prioritised as not all molecules have been fully expanded to depth ten, leading to REACTS exploring different regions of chemical space to generate higher scores for each of the problems. Figure 10-28 also highlights that REACTS routinely traverses greater depths than previous RV-based methods (Ghiandoni, 2019; Patel, 2009; Patel et al., 2009; Wallace, 2016). Finally, it should be noted that in the case of REACTS, due to a better prioritisations of reaction transformations and a smaller reagent pool than DINGOS (Button et al.,

2019), it has allowed REACTS to traverse greater depth where DINGOS was only restricted to synthetic schemes of depth four. This again highlights the REACTS method's strength as an approach to search chemical space as part of a reaction-based de novo design software.


Alectinib
1.0
0.8
0.6
0.4
0.2
0.0
Transformed Hamming Distance


Cariprazine
1.0
0.8
0.6
0.4
0.2
0.0
Transformed Hamming Distance


Osimertinib
1.0
0.8
0.6
0.4
0.2
0.0
Transformed Hamming Distance

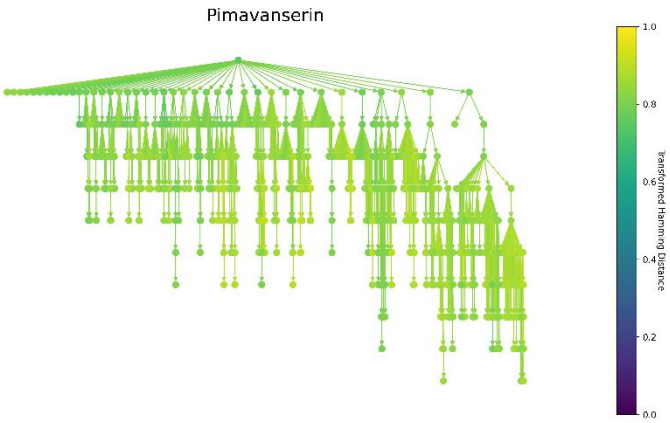


Figure 10-28 Trees Generated by REACTS for the transformed Hamming distance problem. 1) Alectinib, 2)cariprazine,3) osimertinib,4) pimavanserin

Figure 10-29 shows a grid of synthetic tractability metrics for each of the trees that have been generated by REACTS. Again, in all cases, the generated molecules score well against the synthetic tractability metrics. In RAScore, most molecules are deemed to be retro synthetically solvable and therefore likely to be synthesisable. Table 10-16 shows the number of molecules deemed not retrosynthetically solvable based on the RAScore. This would suggest that over 99% of molecules generated by REACTS are retrosynthetic resolvable and thus likely to be synthetically tractable. Finally, Table 10-17 shows the purchasable molecules generated by REACTS for each of the problems; in all cases, molecules were generated by REACTS which could be found directly in the zinc in stock database; however, most of the compounds were not. This could suggest that the majority of molecules generated are novel.

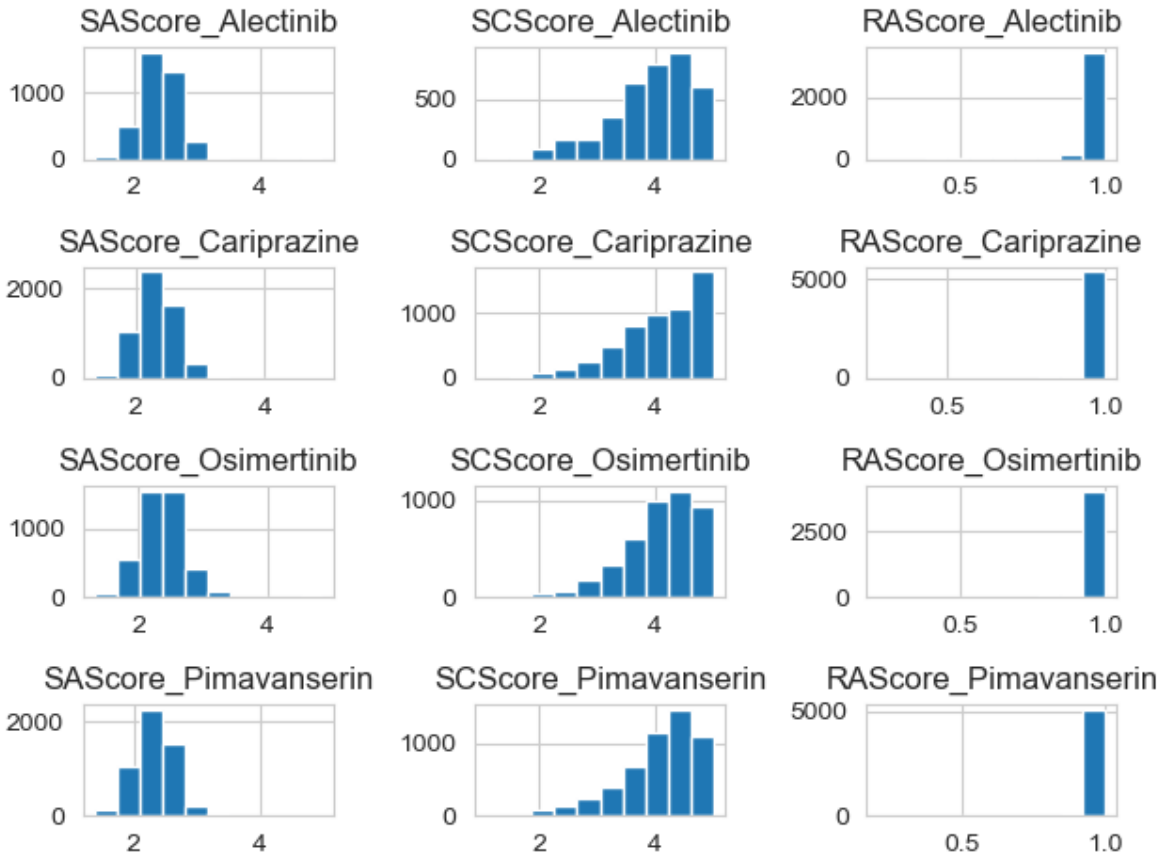


Figure 10-29 grid of SAScores,SCScores, and RAScores of the molecules generated for each tree.

Problem	RAScore (<0.5)
Alectinib	10
Cariprazine	4
Osimertinib	17
Pimaversin	3

Table 10-16 Number of Molecules which were deemed retrosynthetically intractable by RAScore for the REACTS searches

ZINC20-in-stock (tranch)	Purchasable molecules generated by REACTS for the Alectinib problem	Purchasable molecules generated by REACTS for the Cariprazine problem	Purchasable molecules generated by REACTS for the Osimertinib problem	Purchasable molecules generated by REACTS for the Pimavanserin problem
Fragment	32	44	20	31
Lead	0	1	0	0
Drug	2	10	3	10

Table 10-17 Purchasable molecules generated by REACTS for the Transformed hamming loss problems

10.4 Conclusions

This chapter sought to demonstrate that REACTS can be used for a variety of reaction-based de novo design problems. The results from this chapter suggest that REACTS can propose novel synthetically tractable compounds under modest SM and reagent constraints. In every example, REACTS built better compounds than the SM for the corresponding scoring function. In the QED maximisation, the results were on par with established DNN approaches. In the CNS MPO problem, REACTS discovered many solutions at a variety of depths. In the scaffold hopping problem, again, REACTS improved on the SM and proposed compounds which maximised the objective criteria; however, REACTS was limited by the available reagent pool and thus could not be improved further. REACTS was also tested on four transformed Hamming distance problems and compared to the DINGOS software (Button et al., 2019). While REACTS did not find better scoring compounds, the compounds found were ranked competitively to those generated by DINGOS. Remarkably REACTS achieved this with a reagent pool several orders of magnitude smaller than the corresponding DINGOS reagent pool. Moreover, REACTS traversed a much larger transformation pool than DINGOS and to a greater depth.

It is also notable that the paths proposed to the top-scoring compounds generated by REACTS were different from those in other REACTS searches such as the paths explored in the benchmark study, demonstrating that REACTS can prioritise reactions directly based upon the scoring function used and thus chart a path through the chemical space to regions of interest. Finally, a critical consideration that makes REACTS a preferred choice for reaction-based de novo design is the flexibility of the transformation library that can be used. The REACTS algorithm uses a plug and play approach whereby the transformations can be used in domains with large amounts of data or in cases where there are a smaller set of applicable transformations. Hence, compared to DINGOS

which relies on training on a large template pool (Lowe, 2017; Lowe, 2012). REACTS effectively "bootstrapped" the data generation process from the specified reaction vector pool. Being able to self-generate data is a positive benefit as it allows REACTS to be used flexibly with different reaction corpora.

Chapter 11 Conclusions, limitations, and future work

11.1 Conclusions

This thesis has described the development of a novel tree search algorithm for RV-based de novo design. The two fundamental limitations of previous RV-based de novo design algorithms have been the overreliance on complete enumeration search methodologies and the issue of intermediate molecules scoring poorly in reaction paths. The REACTS algorithm developed throughout this thesis has been shown to overcome these limitations on a range of problems. Figure 11-1 below outlines the development, testing, and creation of the REACTS approach.

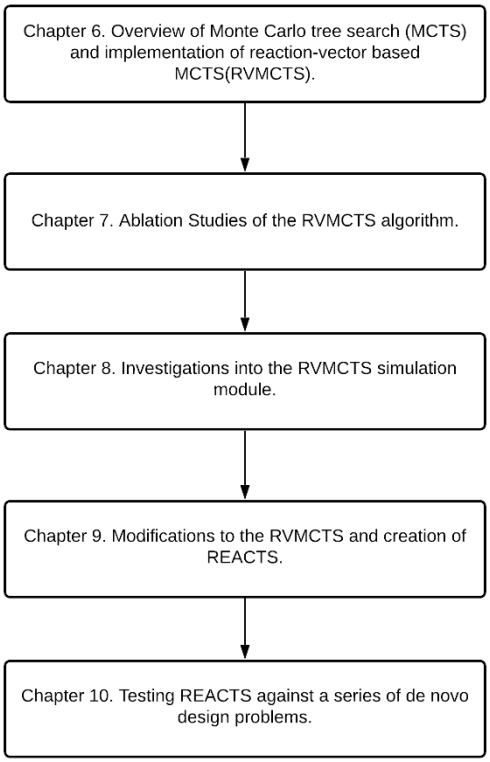


Figure 11-1 Path of the development of REACTS

Chapter 6 overviewed the Monte Carlo tree search (MCTS) algorithm and highlighted both the strengths and weaknesses of the approach. The MCTS was identified as a promising approach to RV-based de novo design. Chapter 6 then details the first RV-based Monte Carlo tree search (RVMCTS) implementation. Next, a validated benchmark was created to test the RVMCTS. A limitation of this validated benchmark is that it contains only three synthetic schemes that could be validated. This does not invalidate the approach of RVs for structure generation. However, it highlights that

cleaning and encoding of reaction schemes containing complex transformations from literature examples is nontrivial.

Chapter 7 investigated the performance of each subcomponent of the RVMCTS via a series of ablation studies. In each experiment, a component of the Monte Carlo tree search was removed. By removal of each component, the corresponding performance of that component could be identified. In particular, the baseline RVMCTS demonstrated good performance on two of the three problems (alogliptin and aripiprazole). The ablation studies demonstrated that removal of selection during RV-based Monte Carlo tree search yielded a random search, which was also true of removal backpropagation. Ablation of expansion improved the overall depth of the trees; however, it also led to greater search instability. The ablation of simulation demonstrated better performance than the baseline, which was attributed to the ablated simulation being a greedier search. In terms of computational resources, the ablation studies showed that using a complete enumeration and a depth one simulation led to inordinate computational expense due to more molecules being generated. Finally, a general robustness measure was explored and demonstrated that ablation of expansion yielded the greatest changes during the search. Chapter 7 demonstrated the need for improved molecule prioritisation beyond the UCT equation. Moreover, the ablation studies highlighted the risk of node removal. However, pruning was identified as a necessity due to the large number of molecules being generated.

Chapter 8 further investigated the use a simulation beyond the ablation studies in Chapter 7. The focus on the simulation was due to the simulation being a novel mechanism not previously explored in reaction-based de novo design. The simulation was investigated via a depth limited random search process. Overall, the results demonstrated a slight performance improvement on the validated benchmark problems. However, this performance came at an extreme computational expense. To overcome the expense, a machine learning approach to simulation search was proposed. This methodology utilised a pre-search data generation methodology coupled with the training of a regression model via large-scale machine-based linear regression. The machine learning regression approach demonstrated equivalent performance as the random simulation with a minimal computational expense. However, a key limitation remained: too many compounds were generated due to reliance on a full enumeration approach during expansion.

Chapter 9 explored several modifications to the RVMCTS. These modifications sought to overcome the limitations identified in Chapter 7. The modifications included improved selection such as Max simulated annealing UCT (Max SA-UCT) and predictor UCT (PUCT). Additionally, several ways of controlling the expansion step were explored such as pruning based on physical-chemical

(PhysChem) descriptors, Local-chemical subspace (LCS) pruning and pruning of RVs using a CB model. Notably, the CB model is a trained single-step reinforcement learning model designed to pick promising RVs. The modifications were then tested utilising the validated benchmark. The results suggested improved selection via Max SA-UCT, PUCT, and expansion pruning utilising LCS and RV pruning using a CB model performed the best. Finally, the MAX-SAUCT and PUCT were combined to form SAPUCT and the SAPUCT, LCS, CB model, and the simulation model from Chapter 8 were combined to form the novel tree search algorithm REACTS (REACtion Tree Search). REACTS, was tested and was able to solve the validated benchmark problems. Notably, REACTS solved the validated benchmark problems even though the intermediates were not ranked highly by the scoring function.

Chapter 10 then tested REACTS on a set of de novo design problems. The problems were maximisation of the quantitative estimation of drug-likeness (Bickerton et al., 2012), central nervous system multiparameter optimisation (Brown et al., 2019; Wager et al., 2010), scaffold hopping (Brown et al., 2019), and finally, four transformed hamming loss problems explored previously by the DINGOS software (Button et al., 2019). A CB model and simulation model was developed for each problem using a generated dataset created from a predefined training set used in the GuacaMol benchmark (Brown et al., 2019). In all problems, REACTS generated molecules better than the SM. Moreover, REACTS demonstrated comparable performance in both the QED and CNS MPO problems to establish DNN methods. However, REACTS did not perform well in the scaffold hopping problem due to a constrained reagent pool. REACTS was finally tested against a set of transformed hamming loss problems. In all cases, REACTS generated molecules of comparable quality to those generated by DINGOS (Button et al., 2019) whilst using a significantly smaller reagent pool. REACTS demonstrated that it could traverse the chemical space and prioritise reactions based on the scoring function. In all cases, the molecules generated by REACTS were deemed synthetically tractable based on in-silico metrics (Coley et al., 2018; Ertl & Schuffenhauer, 2009; Thakkar et al., 2021). Additionally, it was shown that in every search, molecules were generated that were purchasable directly from the vendors. The results suggest REACTS can traverse synthetically tractable chemical space utilising paths much deeper than previous RV-based methodologies.

11.2 Limitations of the work

It should be noted that several limitations of this work have been identified and are valuable avenues for possible future improvement and investigation. The work described in this thesis relied on the use of RVs as a de novo design construction method.

However, RVs have several limitations that influenced the development of this work. The critical limitation is that the RV engine has been optimised and developed for large scale encoding and enumeration of applicable RVs. In addition, the RV is based on the encoding of single-step transformations rather than sequential reactions. As a result, this limitation led to only three reaction schemes being investigated in the benchmark. However, these schemes allowed investigation into the performance of the MCTS via ablation studies.

Integrating RVs with the MCTS led to several design decisions to alleviate the core issue of speed of access to individual RVs and reagents. For example, the generation of new molecules throughout this work heavily emphasised enumeration which meant that there had to be an approach to prioritising RV transformations utilising a post processing methodology. Another limitation of the work is that, only a single RV database was used. This is not a significant limitation but is notable and an avenue for future investigation.

The further limitation of this work is the assessment of the regression methodology. In this work, REACTS utilises a regression-based model to map input and simulation outcome to depth 1. The core limitation of this is that the regression model was assessed via 10-CV-R^2. The 10-CV-R^2 is limited in the effect of outliers and does not consider the learnt model's coefficients directly. It is important to note that whether the regression model will benefit a search can only be assessed when the search is executed. Any prejudgement via a series of metrics assumes that the distribution of molecules seen in the training data will be the same as the molecules at inference time. Furthermore, only linear-based models were explored as a regression method. Linear regression models lack expressiveness, and therefore can be limited on data which is non-linear.

Another limitation is the data generation process to train the regression and CB models. The models developed within this thesis utilised training data from a large dataset that had to be generated, scored, and transformed prior to training. This generation process needs only to occur once but the data must be rescored for each new problem, and the models retrained, and this could lead to a severe time penalty for more computationally demanding scoring functions.

The development and testing of REACTS was based on a set of de novo design rediscovery problems. Although the methods were then tested on different types of problems in Chapter 10 , these

problems were also similarity-based. While this is not a limitation per se, it would have been beneficial to explore alternative problem types such as receptor-based scoring.

The hyper-parameters selected throughout this work for implementing the MCTS and then modified RVMCTS and finally REACTS were selected based upon approximation and knowledge of the RV engine. A more systematic approach to hyperparameter optimisation such as a grid search was not explored due to computational expense. As referenced in Chapter 6, a fundamental limitation is that the ideal hyper-parameters for a given problem will not be known until the search is executed. Therefore a "best guess" approach is required to select default parameters. In Chapter 10, the parameters were chosen based upon the findings of the previous experiments. In future work, it would be interesting to leverage hyperparameter optimisation methods such as Bayesian optimisation or a genetic algorithm to select the best possible search parameters.

Finally, the limitations of the RV approach itself should be acknowledged. REACTS is unable to construct complex heterocyclic ring systems due to limitations of the structure generation approach. It may therefore perform poorly on scaffold hopping problems that require replacing a specific ring system. Additionally, REACTS requires a generated dataset to be compiled to train the simulation model and the CB model. This generated dataset requires parallel computation to score the many molecules that are generated. Finally, although the molecules generated by REACTS using RVs are deemed synthetically tractable using in-silico measures, there is no explicit guarantee that the proposed synthetic routes would be transferable to the lab.

Finally, this work has been carried out over several years. Therefore, it is important to reflect on the progress made in chemoinformatics and de novo design during that time and how developments in the field may have shaped this research differently. In particular, however, on reflection a prospective study utilising scoring functions using "real-world" examples would have provided more utility than perhaps reliance on similarity alone.

Moreover, as the field of reaction prediction has progressed, leveraging reaction transformations utilising data-driven deep neural networks would have been an area that may have guided the research differently. These methods use modern deep neural network architectures such as transformers to predict the outcomes of reactions when provided starting materials and reagents and demonstrate impressive accuracy on benchmarked datasets (Schwaller et al., 2019).

Ultimately, the core limitations of de novo design are still present. Synthetic tractability is still a concern. Interfacing methods with practicing chemists are still difficult. Therefore, the focus would

still be on MCTS as a search tool. However, modifications and validation, would utilise the expertise of end-users.

11.3 Future work

The process of developing and testing de novo design software is a never-ending process. Several limitations have been identified and are promising as directions for future work. First, REACTS leverages machine learning to improve computational efficiency and improve the likelihood of generating high scoring molecules. Machine learning as part of REACTS is used for the simulation model and RV prioritisation. An exciting direction for future work would be further exploration of the large-scale linear regression approach, including more extensive hyper-parameter optimisation and a more exhaustive investigation of different molecular fingerprints. Additionally, artificial neural network approaches could also be investigated for the simulation model. Second, the CB model approach is novel to this work. The CB model approach is a powerful approach to modelling complex single-step decision problems. A Morgan Fingerprint has been used in this work as part of REACTS; however, a future research direction would be investigating alternative molecular fingerprints and better exploring different approaches to training a CB model. REACTS also relies upon local chemical subspaces (LCS) pruning. LCS pruning is an exciting direction for future work as the parameters of the LCS pruning step can be further refined for each de novo design problem. A further direction for future work would be to investigate the usage of self-play generation of training data (Silver et al., 2018), where sequential REACTS searches are used to build a dataset from which the simulation model, CB model, and LCS pruning are iteratively improved. Another direction of future work is directly related to the fundamentals of RV-based de novo design. Improvements to the RV-based structure generation through stereochemistry or better handling of long-range effects that influence the reaction centre would also improve REACTS performance. Additionally, further exploration using a larger reagent and reaction pool to see REACTS performance would be beneficial. Finally, testing REACTS, and the paths proposed by the approach would be exciting as part of a more extensive prospective design study.

Appendix A MCTS worked example

As the MCTS is a multi phase algorithm, it is instructive to complete a worked example.

In advance of the search criteria, there are several definitions which must be completed Here definitions of state, action, evaluation function and stopping criteria are required:

Component	Definition
State	Generic state (S_n)
Action	Transforms a generic state S_n to S_{n+1}
Evaluation function	Takes a Generic State S_n and returns a value of 0 or 1.
Stopping criteria	3 iterations

Here the root is defined:

The node is the only manually defined node that is present within the search.

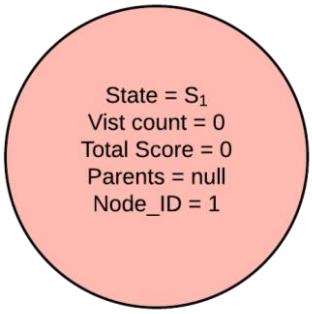


Figure A-1-1 Defining the root node

Iteration one: Selection$_1$

Selection$_1$: Choose which node to expand next based on the UCT equation.

Start at level one and calculate the UCT value for everything.

$$UCT = \frac{Total\ Score}{number\ of\ visits} + 2 * C_p * \sqrt{\frac{2\ln(number\ of\ visits\ to\ parents)}{number\ of\ visits}}$$

$$UCT_{ID1} = \frac{0}{0} + 2 * C_p * \sqrt{\frac{2\ln(NULL)}{0}}$$

$$UCT_{ID1} = Undefined$$

No other choice of node exists at that level, so selection of the root node occurs first.

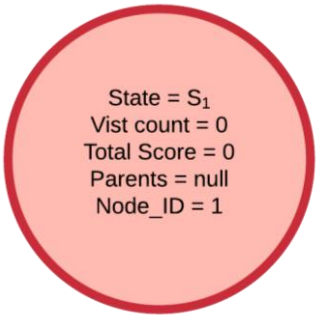


Figure A-1-2 Selecting root node

Expansion$_1$: Take state S_0 and apply an action. Here one action is available which generates two unique states. States cannot exist without nodes, so nodes are generated to store the state and the associated information.

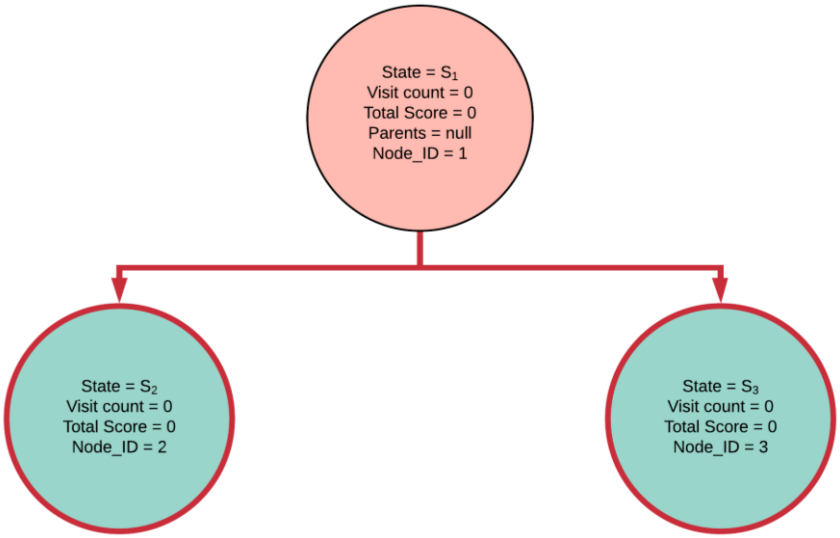


Figure A-1-3 Expanding root node

Simulation$_1$: Assessment of state value needs to occur, so S_2 and S_3 are both simulated. S_2 returns a value (score) of 0 and S_3 returns a value of 1.

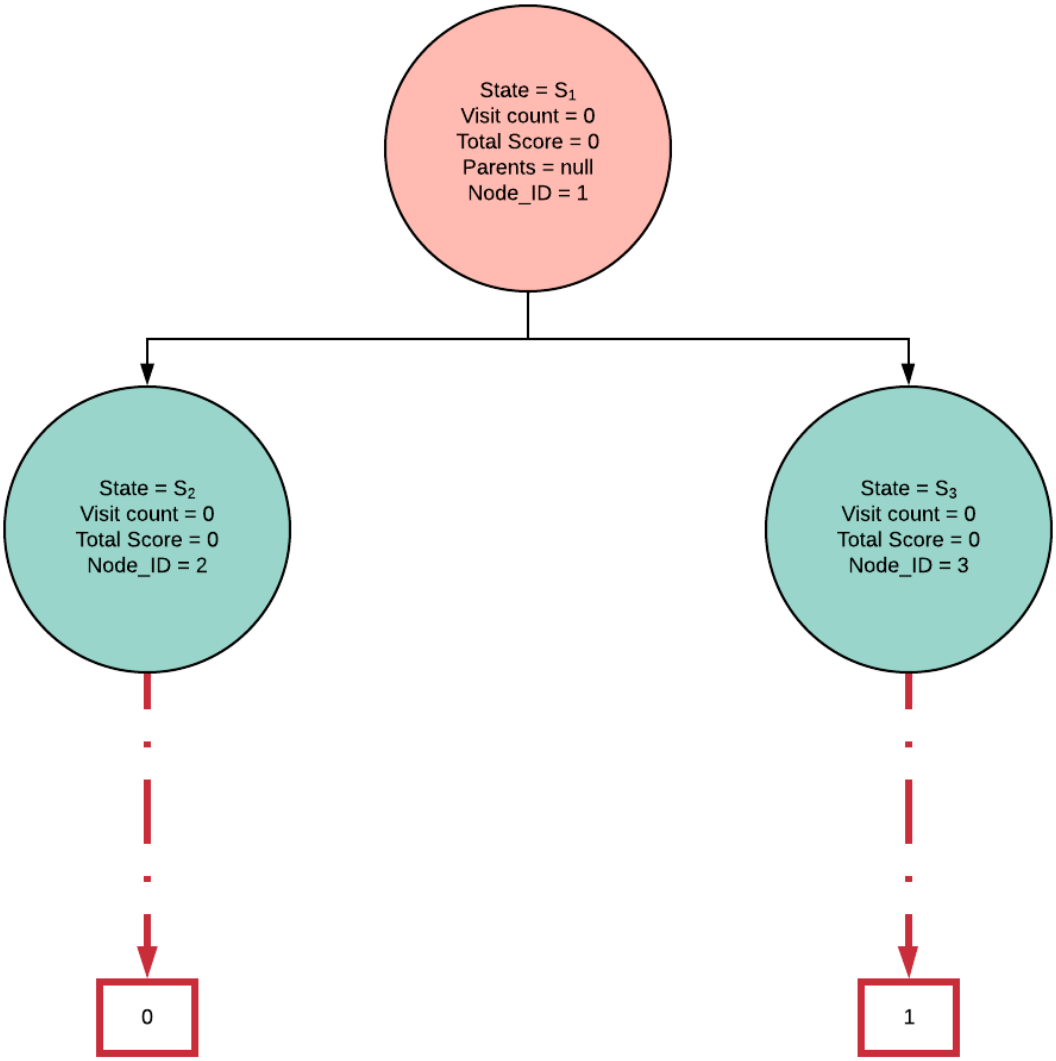


Figure A-1-4 Simulation of expanded nodes

Backpropagation$_1$: Take the simulation value and add it to the score of the node. While also increasing the visit count by 1. The information is then passed up to the parent node. This concludes one iteration, and the algorithm proceeds on to iteration two.

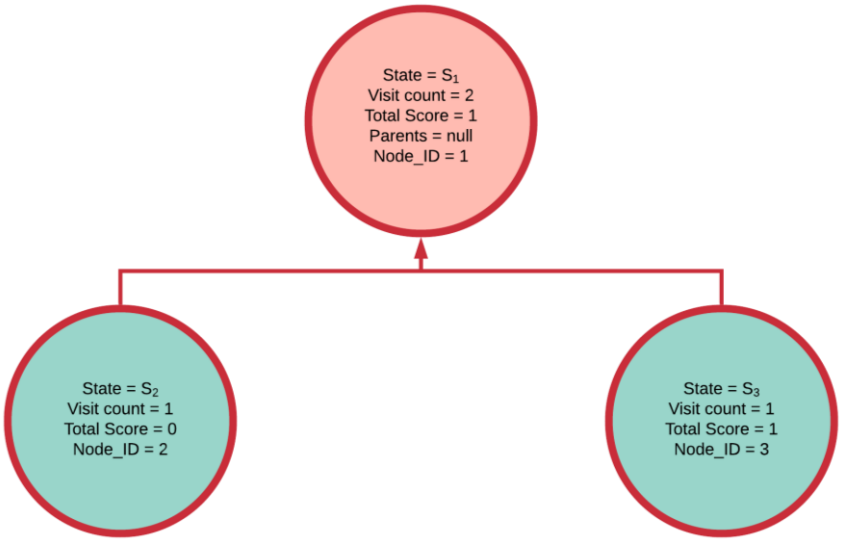


Figure A-1-5 Backpropagation of iteration 1

Hence:

$$Total\ score_2 = 0 + 0 = 0$$

$$Total\ Score_3 = 0 + 1 = 1$$

$$Total\ Score_1 = 0 + 1 = 1$$

$$Visit\ count_2 = 0 + 1 = 1$$

$$Visit\ count_3 = 0 + 1 = 1$$

$$Visit\ count_1 = 1 + 1 = 2$$

Figure A-1-6 Backpropagation of level one nodes

Selection $_2$: Choose which node to expand next based on the UCT equation.

Starting at level one and calculate the UCT value for everything at that level.

$$UCT_{ID1} = \frac{1}{2} + 2 * C_p * \sqrt{\frac{2\ln(NULL)}{2}}$$

$$UCT_{ID1} = Undefined$$

no other choices of node at that level so the root node is selected first.

This time iteration the root node has children so selection proceeds to the Next level

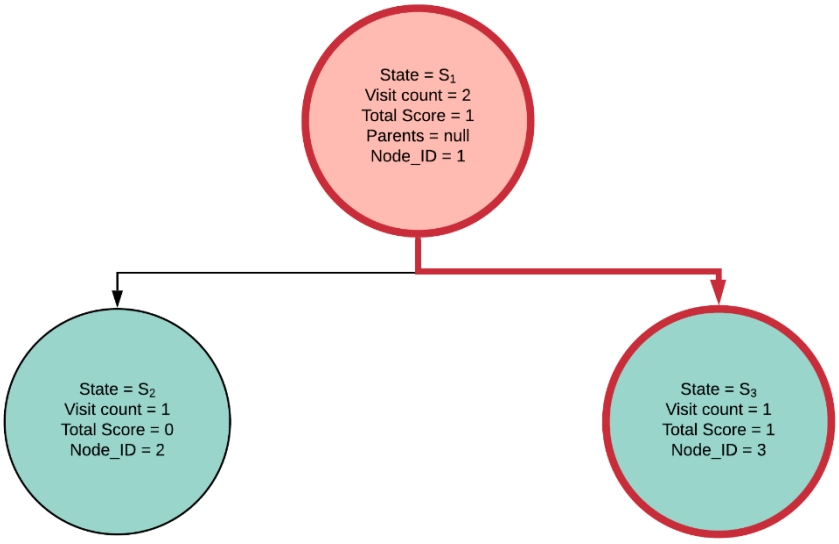


Figure A-1-7 Selection during iteration 2

The UCT score is calculated for nodes at level 2

$$UCT_{ID2} = \frac{0}{1} + 2 * C_p * \sqrt{\frac{2\ln(2)}{1}} = 4.70$$

$$UCT_{ID3} = \frac{1}{1} + 2 * C_p * \sqrt{\frac{2\ln(2)}{1}} = 5.70$$

$$UCT_{ID3} > UCT_{ID2}$$

So Node_ID = 3 is chosen

Expansion$_2$: Take state S_3 and apply an action. Here one action is available which generates two unique states which are stored as newly generated nodes.

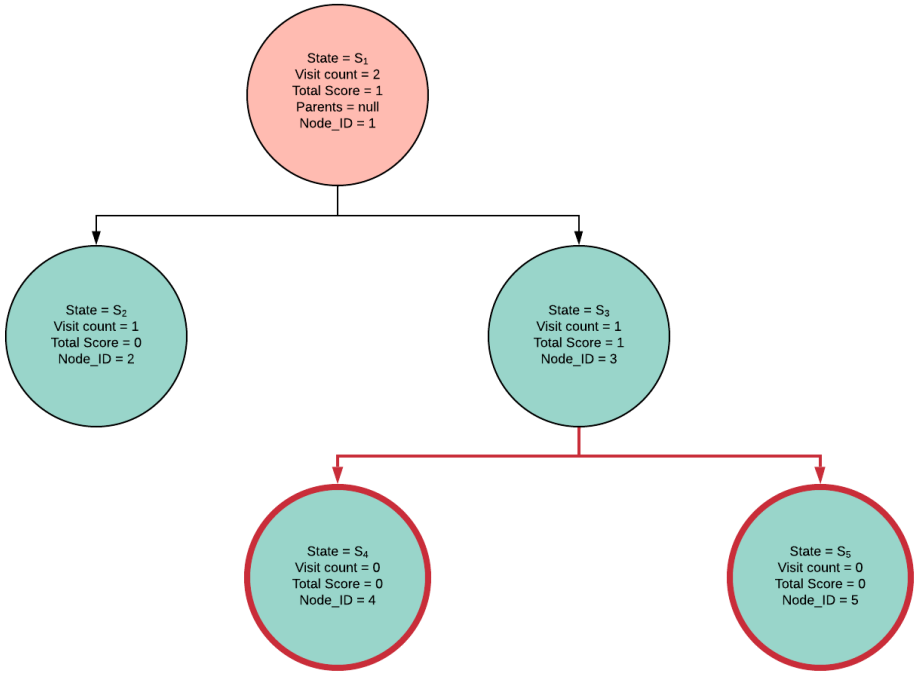


Figure A-1-8 Expansion of level one nodes

Simulation$_2$: Simulating the newly expanded states returns a value of 0 for both of them.

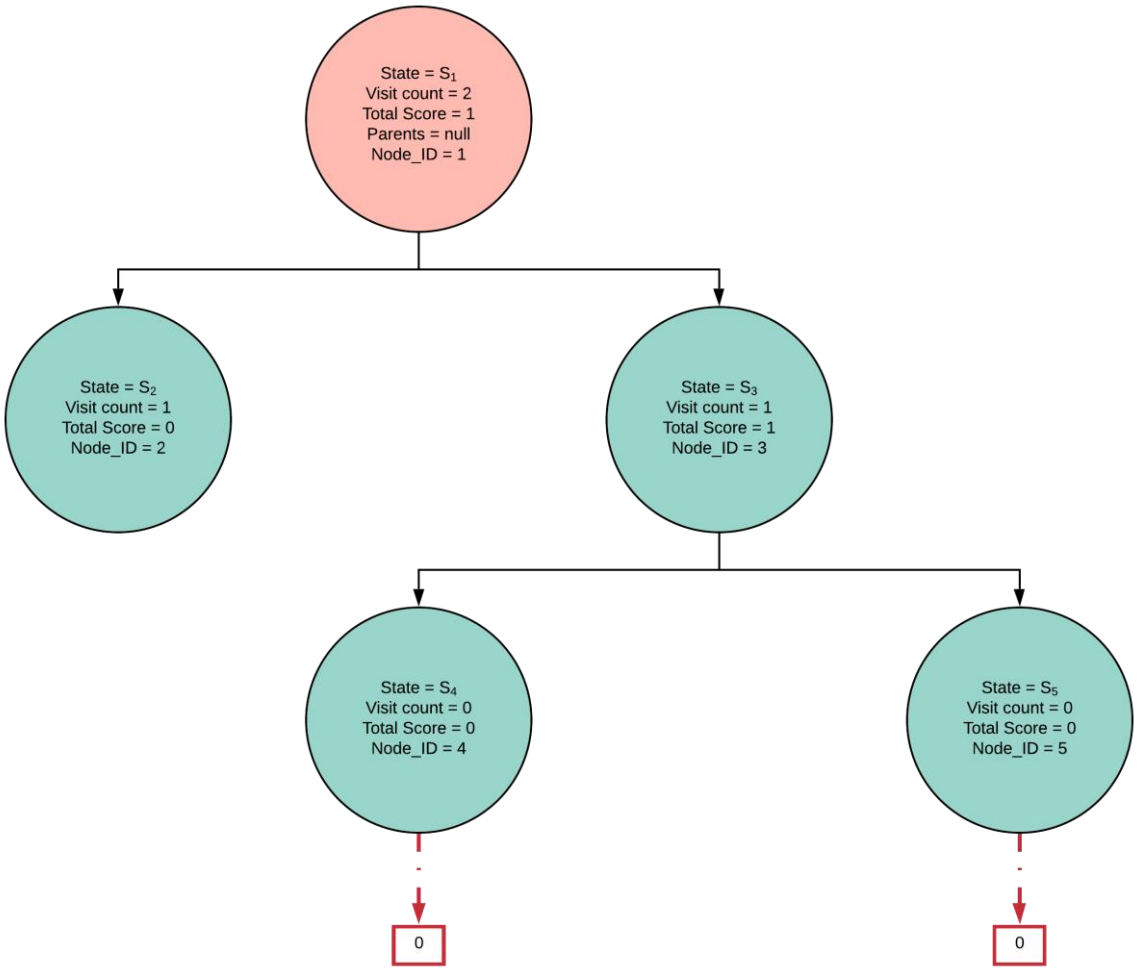


Figure A-1-9 Simulation of new level two nodes

Backpropagation$_2$: The simulation value is taken and added to the score of the node and the visit count is increased by 1. The information is then passed up to the parent node. This concludes one iteration and the proceeds on to iteration three.

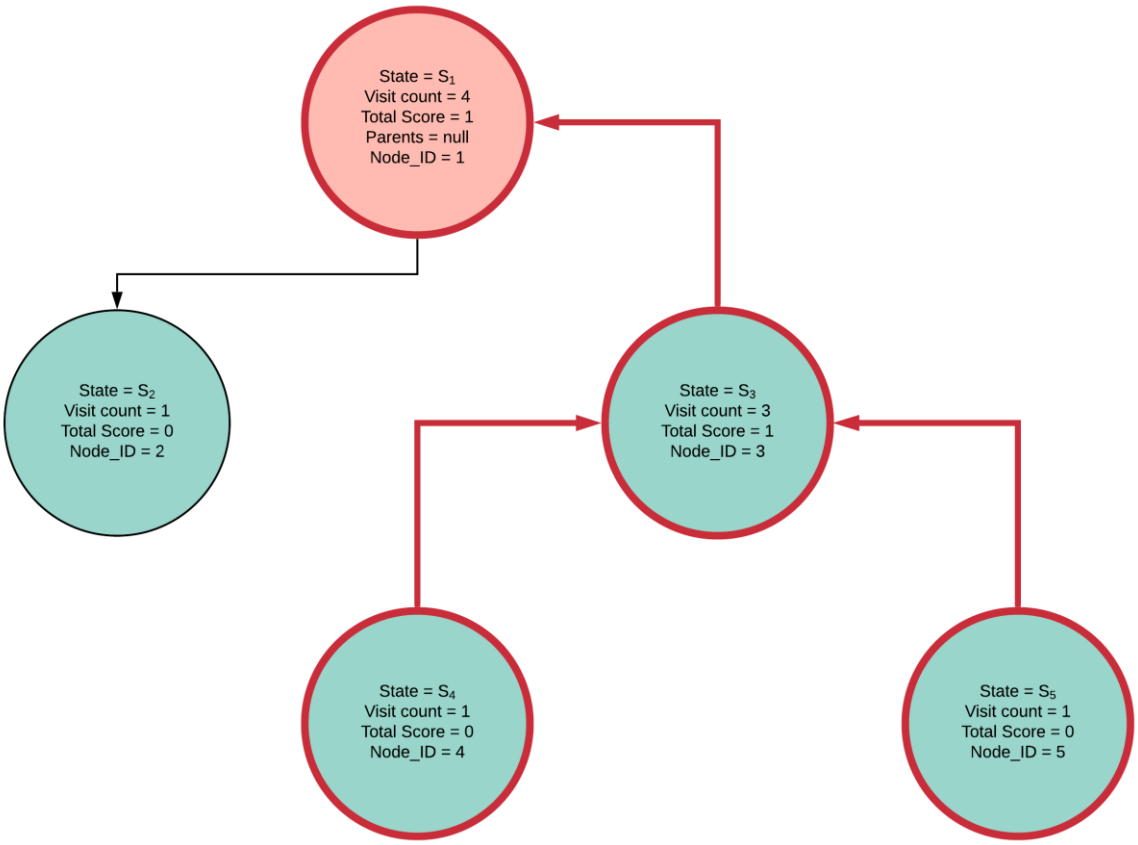


Figure A-1-10 Backpropagation of the newly expanded level two nodes

Hence:

$$Total\ score_4 = 0 + 0 = 0$$

$$Total\ Score_5 = 0 + 0 = 0$$

$$Total\ Score_3 = 0 + 1 = 1$$

$$Total\ Score_1 = 0 + 1 = 1$$

$$Visit\ count_4 = 0 + 1 = 1$$

$$Visit\ count_5 = 0 + 1 = 1$$

$$Visit\ count_3 = 1 + 1 + 1 = 3$$

$$Visit\ count_1 = 2 + 1 + 1 = 4$$

Selection$_3$: Taverse the tree to a leaf based on the UCT score

$$UCT = \frac{Total\ Score}{number\ of\ visits} + 2 * C_p * \sqrt{\frac{2\ln(number\ of\ visits\ to\ parents)}{number\ of\ visits}}$$

Level one: Select Root Node

Level two:

$$UCT_2 = \frac{0}{1} + 2 * c_p * \sqrt{\frac{2\ln(4)}{1}}$$

$$UCT_2 = 6.66$$

$$UCT_3 = \frac{1}{3} + 2 * \mathrm{c_p} * \sqrt{\frac{2\ln(4)}{3}}$$

$$UCT_3 = \frac{1}{3} + 2 * 0.96$$

$$UCT_3 = 4.17$$

$UCT_2 > UCT_3$ Therefore selection of S2.

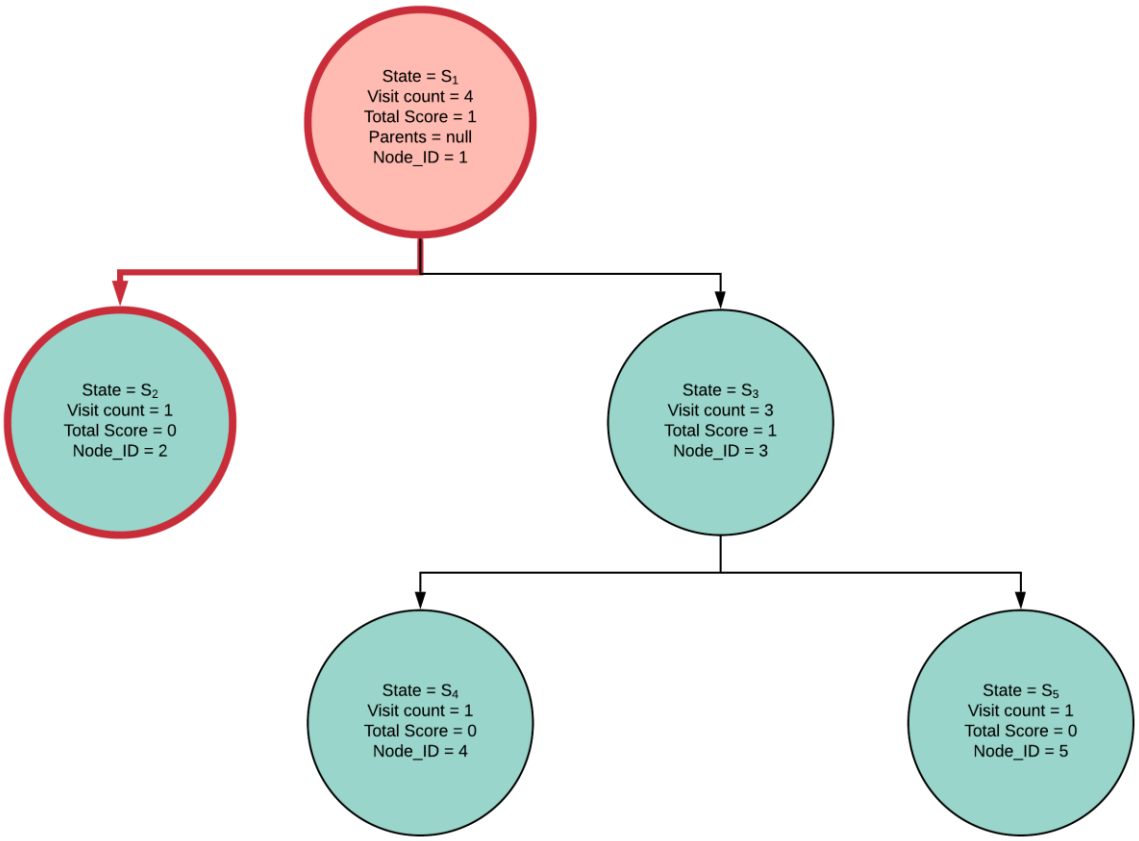


Figure A-1-11 Selection during iteration 3

Expansion$_3$: Take state S_2 and apply an action. Here one action is available which generates one unique state. A node is generated to store the state and the meta information.

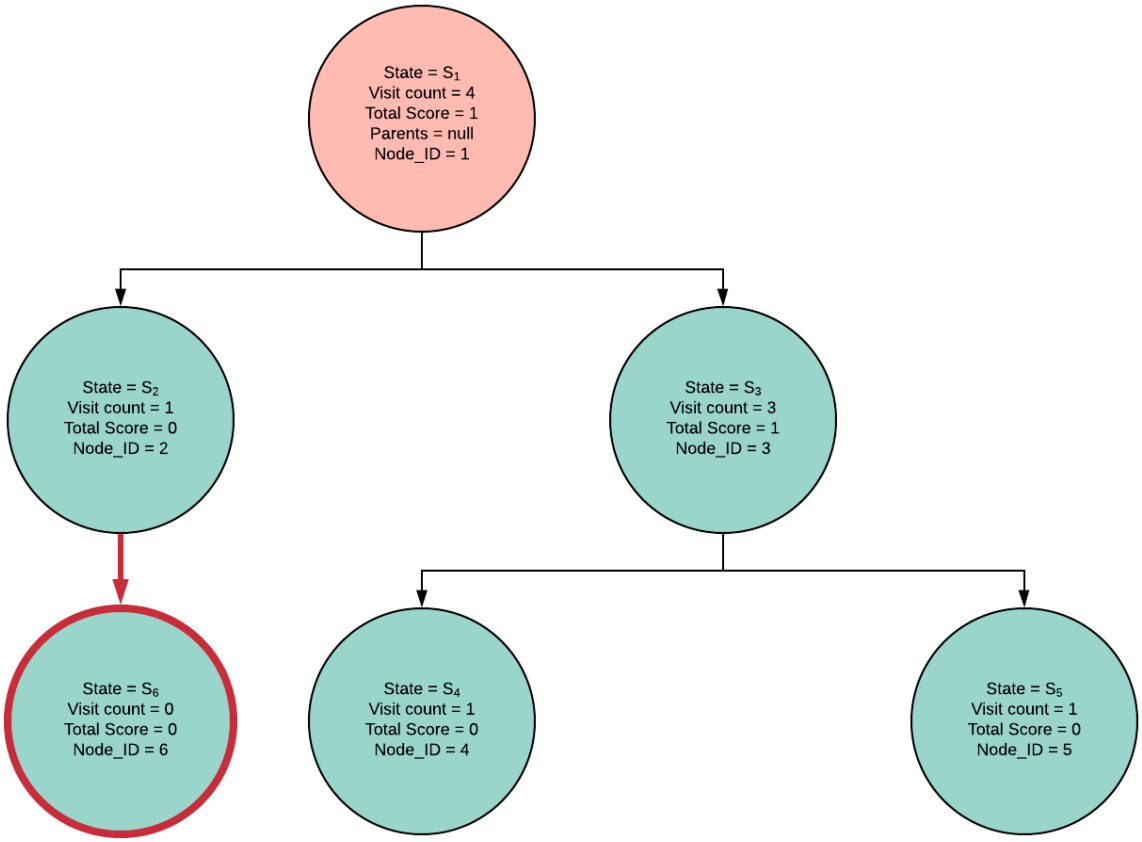


Figure A-1-12 Expansion of S_2

Simulation$_3$: Simulation is performed to assess state value here S_6 returns a value of 1.

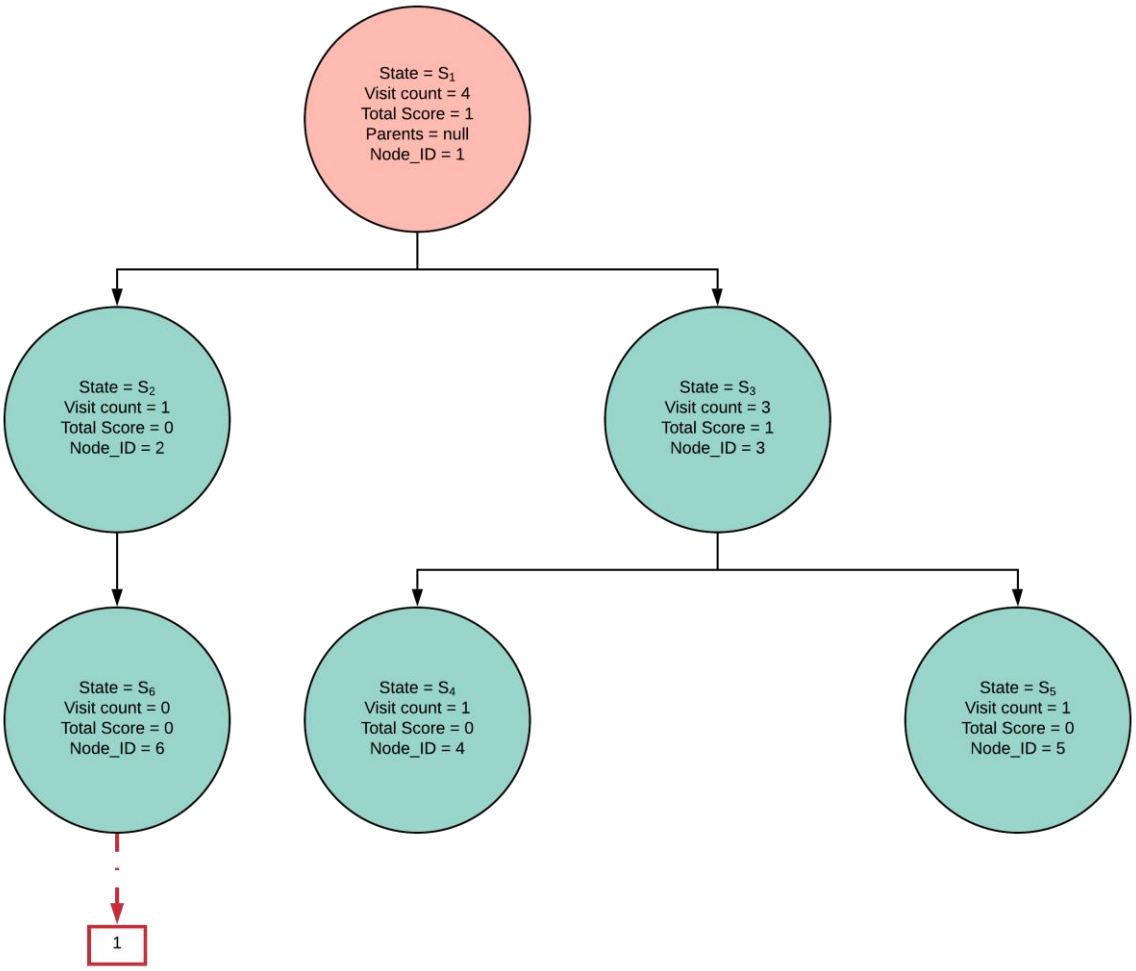


Figure A-1-13 Simulation of newly expanded node

Backpropagation$_3$: The simulation value is taken and added to the score of the node. The visit count is increased by 1. The information is then passed up to the parent node. This concludes three iterations.

Hence:

$$Total\ score_6 = 0 + 1 = 1$$

$$Total\ Score_2 = 0 + 1 = 1$$

$$Total\ Score_1 = 1 + 1 = 2$$

$$Visit\ count_6 = 0 + 1 = 1$$

$$Visit\ count_2 = 1 + 1 = 2$$

$$Visit\ count_1 = 4 + 1 = 5$$

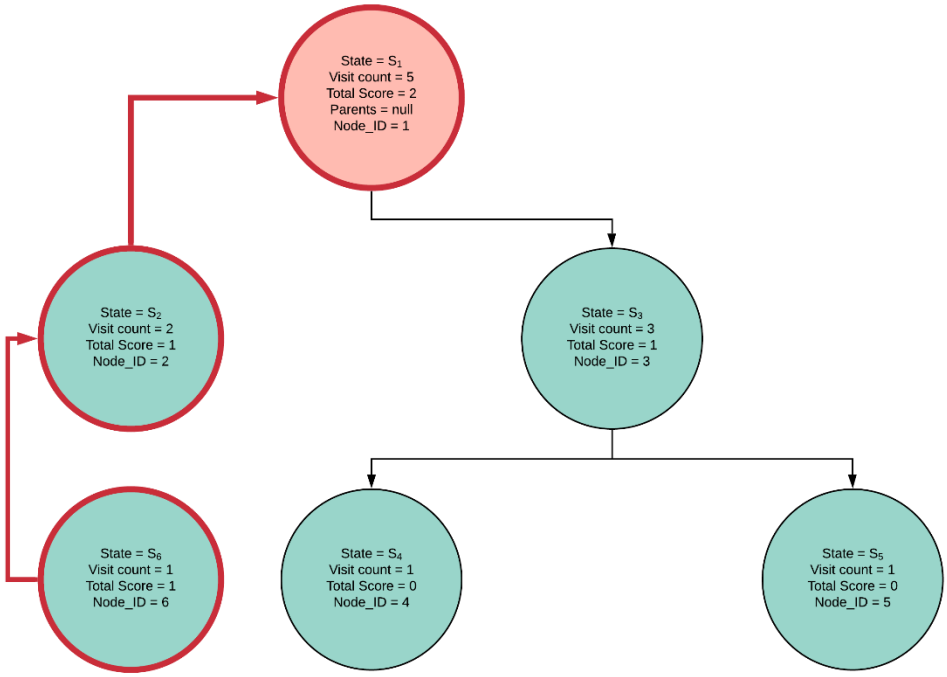


Figure A-1-14 Backpropagation during iteration 3

It is common following the completed MCTS that a final post search "best path" extraction is performed. This is the path with the highest UCT score at each level.

Path Extraction

Level one: Select the root Node S_1.

Level two: $UCT = \frac{Total\ Score}{number\ of\ visits} + 2 * C_p * \sqrt{\frac{2\ln(number\ of\ visits\ to\ parents)}{number\ of\ visits}}$

$$UCT_2 = \frac{1}{2} + 2 * C_p * \sqrt{\frac{2\ln(5)}{2}}$$

$$UCT_2 = 5.57$$

$$UCT_3 = \frac{1}{3} + 2 * C_p * \sqrt{\frac{2\ln(5)}{3}}$$

$$UCT_3 = 4.47$$

Select S2

Level3: one choice S6

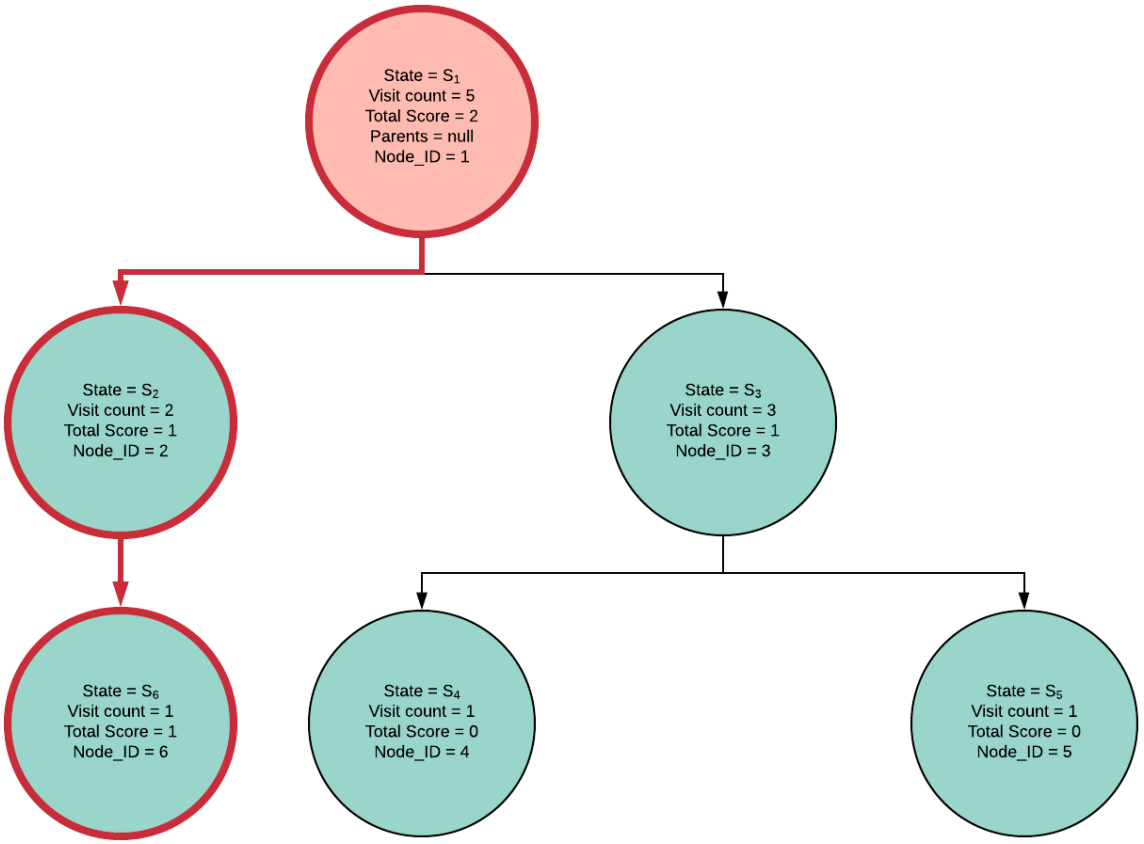


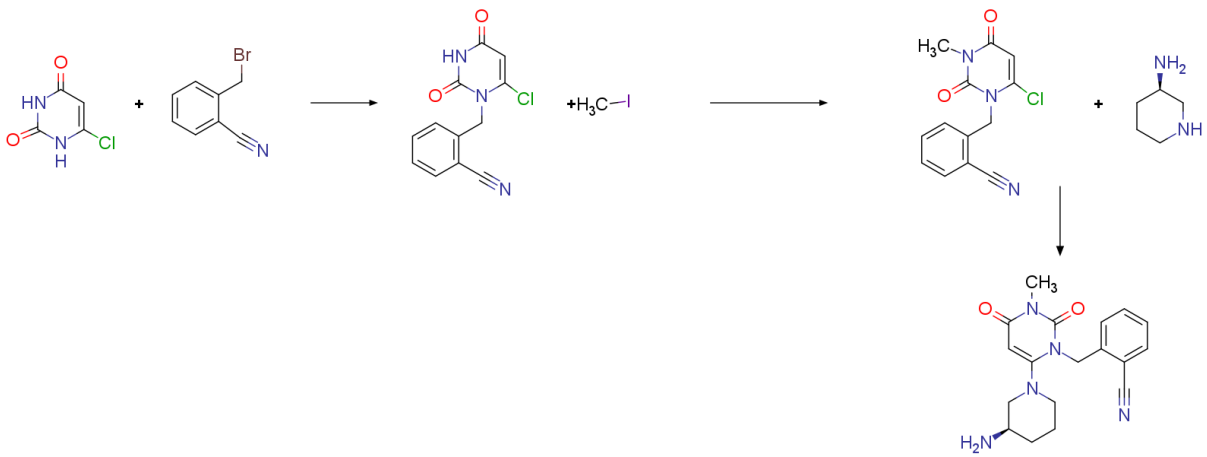
Figure A-1-15 Final path extraction

Final path S1 -> S2 -> S6

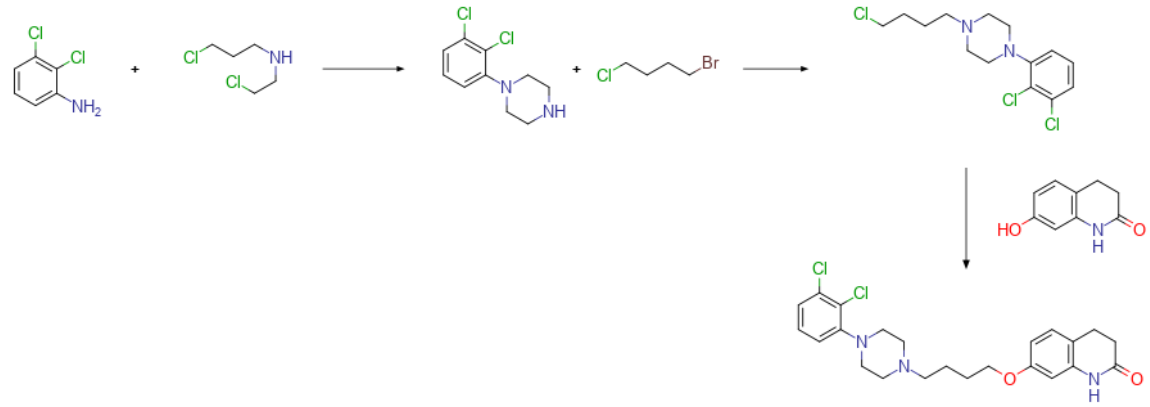
Consequently this finishes all parts of the MCTS and thus the search terminates. Again depending on the domain in question termination may return the entire tree, a sub tree, or only a best performing move.

Appendix B Encoded synthetic schemes

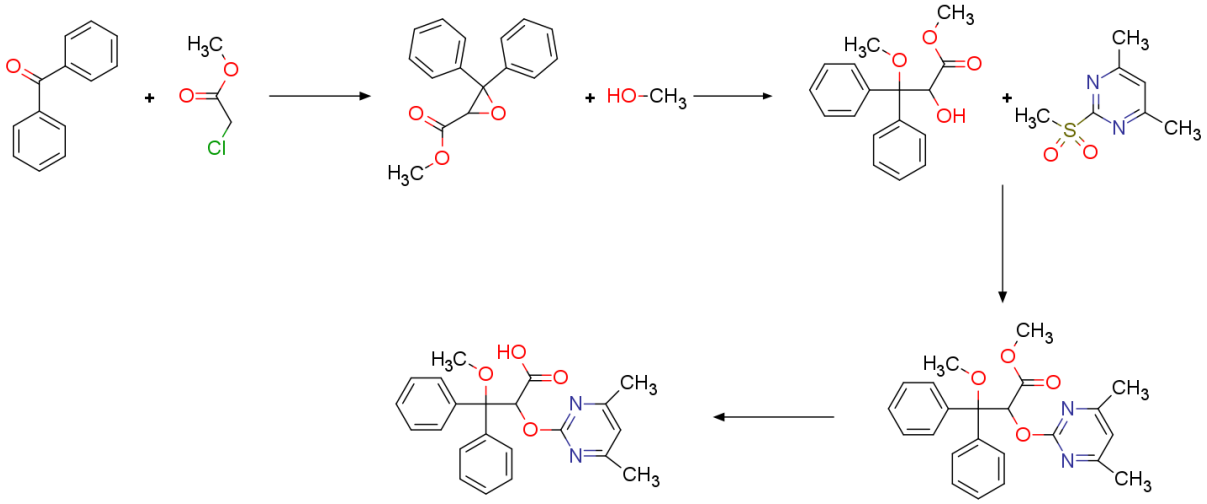
B-1 Alogliptin Synthetic Scheme



B-2 Aripiprazole Synthetic Scheme



B-3 Ambrisentan Synthetic Scheme



Appendix C Ablation studies calculated data

C-1 Ablation studies calculated Max Sim (Tree) (Mean), Max Sim (Tree) (Standard deviation), Average (Tree) (Mean), Average (Tree) (Standard deviation)

Name	Experiment	Max (Tree) (Mean)	Max (Tree) (Standard deviation)	Max (Tree+Sim) (Mean)	Max (Tree+Sim) (Standard deviation)
Alogliptin	Baseline	0.737	0.000	1.000	0.000
Aripiprazole	Baseline	0.468	0.047	0.733	0.051
Ambrisentan	Baseline	0.333	0.000	0.447	0.000
Alogliptin	Ablation of Selection	0.597	0.124	0.842	0.136
Aripiprazole	Ablation of Selection	0.614	0.095	0.603	0.075
Ambrisentan	Ablation of Selection	0.346	0.035	0.820	0.104
Alogliptin	Ablation of Expansion (75% retained)	0.703	0.071	0.948	0.111
Aripiprazole	Ablation of Expansion (75% retained)	0.484	0.045	0.764	0.071
Ambrisentan	Ablation of Expansion (75% retained)	0.333	0.000	0.458	0.038
Alogliptin	Ablation of Expansion (50% retained)	0.588	0.119	0.785	0.161
Aripiprazole	Ablation of Expansion (50% retained)	0.482	0.050	0.754	0.067
Ambrisentan	Ablation of Expansion (50% retained)	0.349	0.051	0.503	0.154
Alogliptin	Ablation of Expansion (25% retained)	0.507	0.187	0.675	0.181
Aripiprazole	Ablation of Expansion (25% retained)	0.437	0.061	0.695	0.080
Ambrisentan	Ablation of Expansion (25% retained)	0.327	0.043	0.518	0.125
Alogliptin	Ablation of Simulation	1.000	0.000	1.000	0.000
Aripiprazole	Ablation of Simulation	0.820	0.000	0.820	0.000
Ambrisentan	Ablation of Simulation	0.352	0.000	0.352	0.000
Alogliptin	Ablation of Backpropagation	0.570	0.062	0.779	0.080
Aripiprazole	Ablation of Backpropagation	0.577	0.083	0.766	0.056
Ambrisentan	Ablation of Backpropagation	0.367	0.054	0.647	0.105

C-2 Ablation studies calculated Average (Tree) (Mean), Average (Tree) (Standard deviation),Average (Tree+Sim) (Mean), Average (Tree+Sim) (Standard deviation)

Name	Experiment	Average (Tree) (Mean)	Average (Tree) (Standard deviation)	Average (Tree+Sim) (Mean)	Average (Tree+Sim) (Standard deviation)
Alogliptin	Baseline	0.142	0.008	0.130	0.008
Aripiprazole	Baseline	0.169	0.030	0.155	0.016
Ambrisentan	Baseline	0.176	0.002	0.160	0.002
Alogliptin	Ablation of Selection	0.113	0.007	0.108	0.005
Aripiprazole	Ablation of Selection	0.154	0.006	0.148	0.005
Ambrisentan	Ablation of Selection	0.170	0.003	0.153	0.003
Alogliptin	Ablation of Expansion (75% retained)	0.143	0.009	0.138	0.009
Aripiprazole	Ablation of Expansion (75% retained)	0.201	0.038	0.171	0.022
Ambrisentan	Ablation of Expansion (75% retained)	0.189	0.002	0.161	0.002
Alogliptin	Ablation of Expansion (50% retained)	0.137	0.021	0.132	0.019
Aripiprazole	Ablation of Expansion (50% retained)	0.191	0.035	0.170	0.022
Ambrisentan	Ablation of Expansion (50% retained)	0.185	0.002	0.161	0.002
Alogliptin	Ablation of Expansion (25% retained)	0.150	0.033	0.142	0.027
Aripiprazole	Ablation of Expansion (25% retained)	0.195	0.045	0.170	0.025
Ambrisentan	Ablation of Expansion (25% retained)	0.180	0.002	0.158	0.002
Alogliptin	Ablation of Simulation	0.484	0.004	0.080	0.000
Aripiprazole	Ablation of Simulation	0.511	0.000	0.180	0.000
Ambrisentan	Ablation of Simulation	0.222	0.000	0.210	0.000
Alogliptin	Ablation of Backpropagation	0.109	0.005	0.108	0.005
Aripiprazole	Ablation of Backpropagation	0.160	0.005	0.147	0.005
Ambrisentan	Ablation of Backpropagation	0.179	0.003	0.152	0.003

C-3 Ablation studies calculated Skewness (Tree) (Mean), Skewness (Tree) (Standard deviation), Kurtosis (Tree) (Mean), Kurtosis (Tree) (Standard deviation)

Name	Experiment	Skewness (Tree) (Mean)	Skewness (Tree) (Standard deviation)	Kurtosis (Tree) (Mean)	Kurtosis (Tree) (Standard deviation)
Alogliptin	Baseline	2.906	0.410	11.541	3.702
Aripiprazole	Baseline	0.269	0.255	2.105	2.404
Ambrisentan	Baseline	-0.151	0.037	-0.151	0.037
Alogliptin	Ablation of Selection	1.923	0.896	13.637	7.406
Aripiprazole	Ablation of Selection	0.714	0.268	3.759	0.699
Ambrisentan	Ablation of Selection	-0.656	0.224	-0.656	0.224
Alogliptin	Ablation of Expansion (75% retained)	2.544	0.322	9.454	2.700
Aripiprazole	Ablation of Expansion (75% retained)	0.504	0.392	2.002	3.711
Ambrisentan	Ablation of Expansion (75% retained)	-0.892	0.085	-0.892	0.085
Alogliptin	Ablation of Expansion (50% retained)	2.456	0.697	9.170	5.265
Aripiprazole	Ablation of Expansion (50% retained)	0.878	0.587	2.707	3.114
Ambrisentan	Ablation of Expansion (50% retained)	-0.709	0.097	-0.709	0.097
Alogliptin	Ablation of Expansion (25% retained)	1.446	0.934	4.169	4.399
Aripiprazole	Ablation of Expansion (25% retained)	0.582	0.595	2.055	2.717
Ambrisentan	Ablation of Expansion (25% retained)	-0.598	0.176	-0.598	0.176
Alogliptin	Ablation of Simulation	-1.175	0.039	2.698	0.016
Aripiprazole	Ablation of Simulation	-1.163	0.003	0.730	0.009
Ambrisentan	Ablation of Simulation	-1.639	0.010	-1.639	0.010
Alogliptin	Ablation of Backpropagation	2.092	0.416	11.889	4.228
Aripiprazole	Ablation of Backpropagation	0.967	0.343	5.329	0.965
Ambrisentan	Ablation of Backpropagation	-1.042	0.189	-1.042	0.189

C-4 Ablation studies calculated Depth (Mean), Depth (Standard deviation), Pairwise similarity (Mean), Pairwise similarity (Standard deviation)

Name	Experiment	Depth (Mean)	Depth (Standard deviation)	Pairwise similarity (Mean)	Pairwise similarity (Standard deviation)
Alogliptin	Baseline	3.000	0.000	0.236	0.007
Aripiprazole	Baseline	2.100	0.316	0.236	0.014
Ambrisentan	Baseline	2.000	0.000	0.247	0.003
Alogliptin	Ablation of Selection	3.100	0.316	0.226	0.009
Aripiprazole	Ablation of Selection	3.000	0.000	0.213	0.007
Ambrisentan	Ablation of Selection	3.000	0.000	0.236	0.009
Alogliptin	Ablation of Expansion (75% retained)	3.000	0.000	0.229	0.014
Aripiprazole	Ablation of Expansion (75% retained)	2.500	0.527	0.245	0.020
Ambrisentan	Ablation of Expansion (75% retained)	2.000	0.000	0.252	0.003
Alogliptin	Ablation of Expansion (50% retained)	3.900	0.738	0.223	0.014
Aripiprazole	Ablation of Expansion (50% retained)	2.500	0.527	0.231	0.015
Ambrisentan	Ablation of Expansion (50% retained)	2.000	0.000	0.245	0.003
Alogliptin	Ablation of Expansion (25% retained)	4.100	1.101	0.229	0.006
Aripiprazole	Ablation of Expansion (25% retained)	5.200	2.044	0.238	0.027
Ambrisentan	Ablation of Expansion (25% retained)	2.500	0.527	0.247	0.007
Alogliptin	Ablation of Simulation	5.000	0.000	0.397	0.003
Aripiprazole	Ablation of Simulation	5.000	0.000	0.388	0.000
Ambrisentan	Ablation of Simulation	3.000	0.000	0.273	0.002
Alogliptin	Ablation of Backpropagation	3.000	0.000	0.228	0.005
Aripiprazole	Ablation of Backpropagation	3.000	0.000	0.216	0.006
Ambrisentan	Ablation of Backpropagation	3.000	0.000	0.231	0.009

C-5 Ablation studies calculated Size (Tree) (Mean), Size (Tree) (Standard deviation), Size (Tree+Sim) (Mean),Size (Tree+Sim) (Standard deviation)

Name	Experiment	Size (Tree) (Mean)	Size (Tree) (Standard deviation)	Size (Tree+Sim) (Mean)	Size (Tree+Sim) (Standard deviation)
Alogliptin	Baseline	866.100	177.271	86270.000	46298.409
Aripiprazole	Baseline	1062.900	302.375	264499.900	153183.233
Ambrisentan	Baseline	9130.800	165.694	965940.600	97320.132
Alogliptin	Ablation of Selection	6906.400	845.517	1067181.700	290110.369
Aripiprazole	Ablation of Selection	15317.200	796.160	3621257.000	452350.526
Ambrisentan	Ablation of Selection	14745.700	992.626	3080665.300	433735.504
Alogliptin	Ablation of Expansion (75% retained)	623.200	242.227	56474.900	25109.768
Aripiprazole	Ablation of Expansion (75% retained)	1097.900	618.610	186767.800	84808.507
Ambrisentan	Ablation of Expansion (75% retained)	7018.600	114.955	935339.500	65773.497
Alogliptin	Ablation of Expansion (50% retained)	516.400	219.960	52140.100	24231.762
Aripiprazole	Ablation of Expansion (50% retained)	1803.000	1323.922	270504.300	183689.258
Ambrisentan	Ablation of Expansion (50% retained)	4910.700	120.686	801707.900	65240.402
Alogliptin	Ablation of Expansion (25% retained)	272.000	133.138	27436.900	13870.227
Aripiprazole	Ablation of Expansion (25% retained)	607.500	324.920	89616.900	46211.560
Ambrisentan	Ablation of Expansion (25% retained)	2850.500	73.773	600588.900	24435.684
Alogliptin	Ablation of Simulation	6590.900	211.925	1.000	0.000
Aripiprazole	Ablation of Simulation	8245.800	17.517	1.000	0.000
Ambrisentan	Ablation of Simulation	13197.200	145.251	1.000	0.000
Alogliptin	Ablation of Backpropagation	7324.500	1090.210	1097119.300	255355.518
Aripiprazole	Ablation of Backpropagation	14701.900	1075.722	3450424.000	512507.722
Ambrisentan	Ablation of Backpropagation	13717.300	486.074	2791764.600	145833.195

C-6 Ablation studies calculated Branching Factor (Mean), Branching Factor (Standard deviation), Time (Mean), Time (Standard deviation)

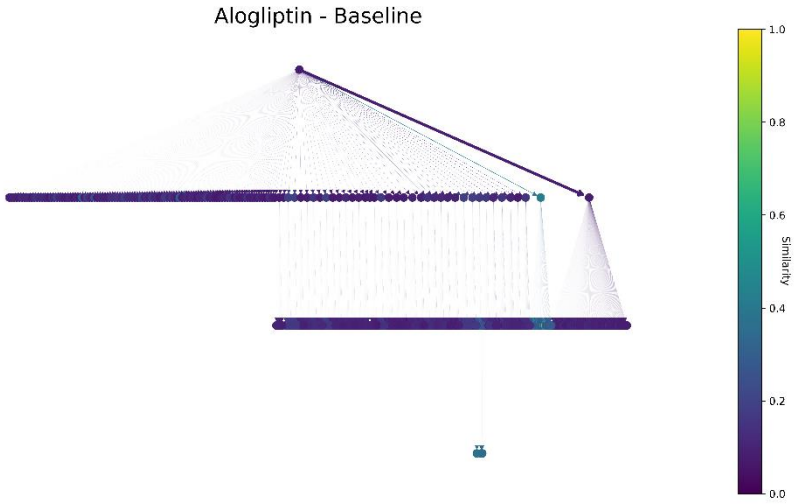
Name	Experiment	Branching Factor (Mean)	Branching Factor (Standard deviation)	Time (Mean)	Time (Standard deviation)
Alogliptin	Baseline	20.066	4.081	14.190	8.263
Aripiprazole	Baseline	166.124	42.030	39.955	26.543
Ambrisentan	Baseline	179.016	3.249	157.683	18.015
Alogliptin	Ablation of Selection	136.184	16.445	185.902	53.780
Aripiprazole	Ablation of Selection	301.460	14.609	579.311	79.342
Ambrisentan	Ablation of Selection	289.112	19.463	520.857	79.463
Alogliptin	Ablation of Expansion (75% retained)	16.092	3.482	9.221	4.026
Aripiprazole	Ablation of Expansion (75% retained)	99.193	63.372	27.960	13.821
Ambrisentan	Ablation of Expansion (75% retained)	137.600	2.254	156.657	11.508
Alogliptin	Ablation of Expansion (50% retained)	13.115	3.185	8.681	4.232
Aripiprazole	Ablation of Expansion (50% retained)	66.898	24.369	43.058	29.812
Ambrisentan	Ablation of Expansion (50% retained)	96.269	2.366	131.917	11.998
Alogliptin	Ablation of Expansion (25% retained)	7.916	1.494	4.708	2.472
Aripiprazole	Ablation of Expansion (25% retained)	20.497	13.525	14.168	7.497
Ambrisentan	Ablation of Expansion (25% retained)	55.873	1.447	99.906	3.923
Alogliptin	Ablation of Simulation	129.214	4.155	1.564	0.047
Aripiprazole	Ablation of Simulation	161.663	0.343	1.736	0.034
Ambrisentan	Ablation of Simulation	258.749	2.848	2.636	0.038
Alogliptin	Ablation of Backpropagation	143.891	21.439	188.980	44.541
Aripiprazole	Ablation of Backpropagation	288.781	20.476	563.790	77.628
Ambrisentan	Ablation of Backpropagation	268.947	9.531	476.100	28.463

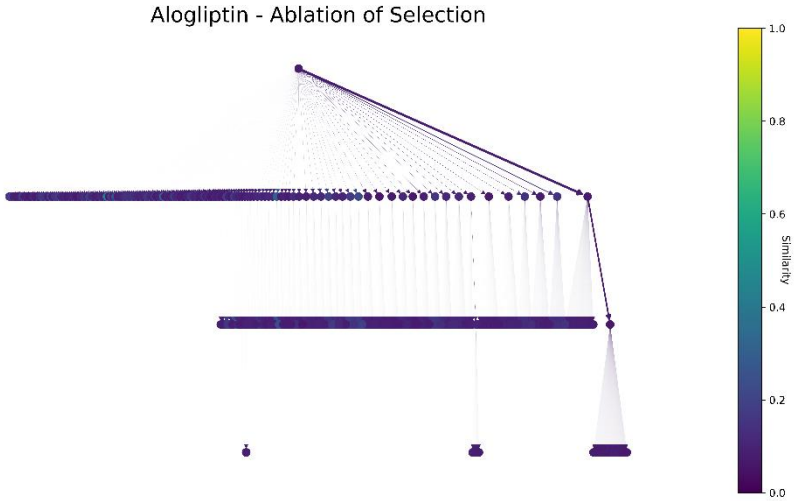
C-7 Ablation studies calculated SDD (Mean), Unique RVs Expanded (Mean),Unique RVs Expanded (Standard deviation)

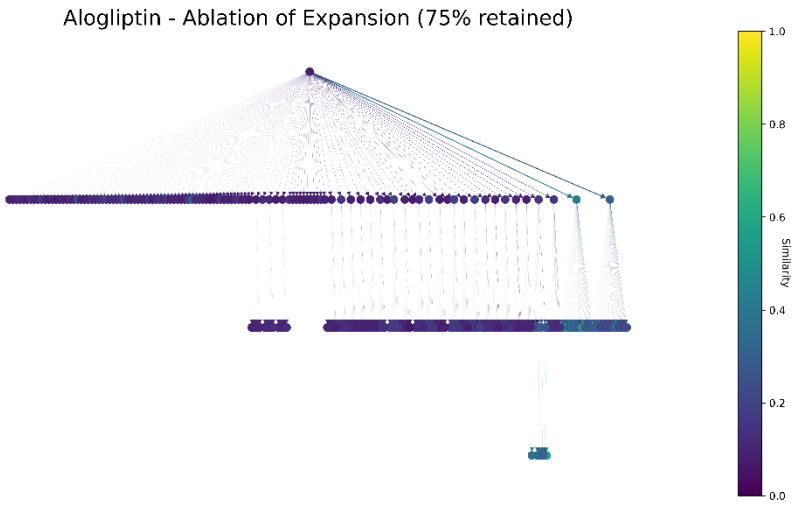
Name	Experiment	SDD (Mean)	Unique RVs Expanded (Mean)	Unique RVs Expanded (Standard deviation)
Alogliptin	Baseline	0.086	2.518	0.477
Aripiprazole	Baseline	0.066	2.114	0.690
Ambrisentan	Baseline	0.048	2.453	0.293
Alogliptin	Ablation of Selection	0.045	13.660	1.276
Aripiprazole	Ablation of Selection	0.057	15.446	1.473
Ambrisentan	Ablation of Selection	0.048	13.192	1.638
Alogliptin	Ablation of Expansion (75% retained)	0.081	2.047	0.851
Aripiprazole	Ablation of Expansion (75% retained)	0.078	2.059	0.561
Ambrisentan	Ablation of Expansion (75% retained)	0.041	2.496	0.269
Alogliptin	Ablation of Expansion (50% retained)	0.075	2.110	0.810
Aripiprazole	Ablation of Expansion (50% retained)	0.069	2.065	0.569
Ambrisentan	Ablation of Expansion (50% retained)	0.044	2.600	0.255
Alogliptin	Ablation of Expansion (25% retained)	0.075	1.462	0.549
Aripiprazole	Ablation of Expansion (25% retained)	0.070	1.697	0.559
Ambrisentan	Ablation of Expansion (25% retained)	0.044	3.390	0.616
Alogliptin	Ablation of Simulation	0.140	8.461	0.393
Aripiprazole	Ablation of Simulation	0.166	6.832	0.000
Ambrisentan	Ablation of Simulation	0.050	8.807	0.000
Alogliptin	Ablation of Backpropagation	0.042	14.772	2.264
Aripiprazole	Ablation of Backpropagation	0.046	14.091	2.340
Ambrisentan	Ablation of Backpropagation	0.044	11.908	2.225

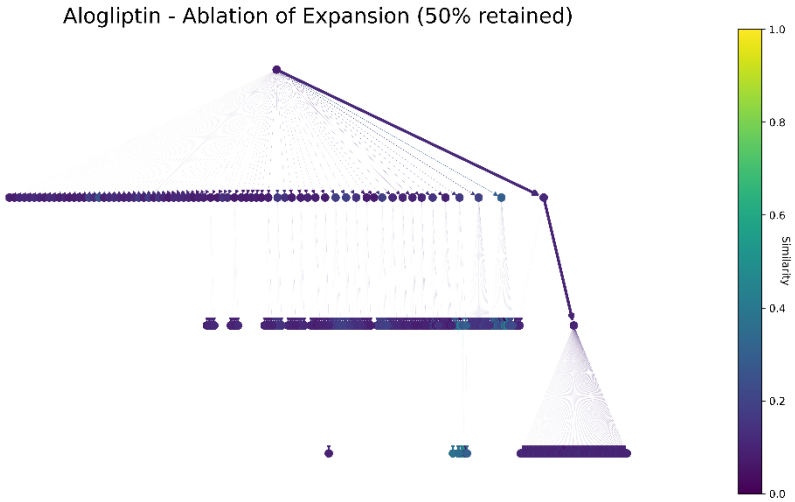
Appendix D Ablation studies selected trees

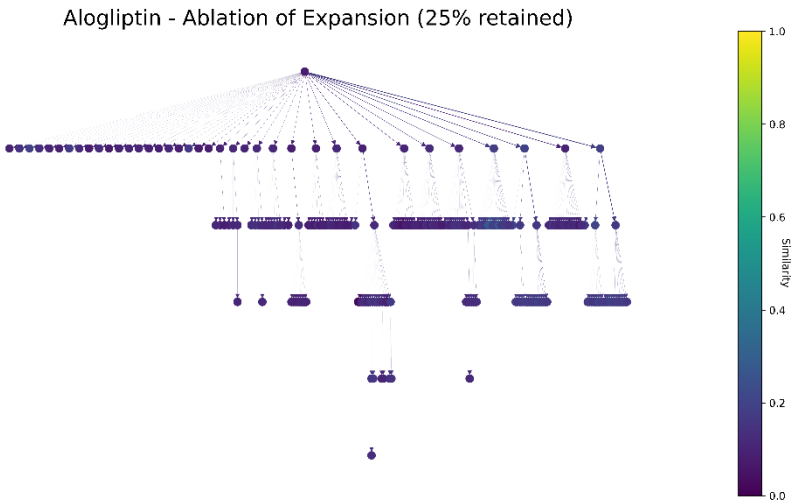
D-1 Selected Alogliptin trees (ablation studies)

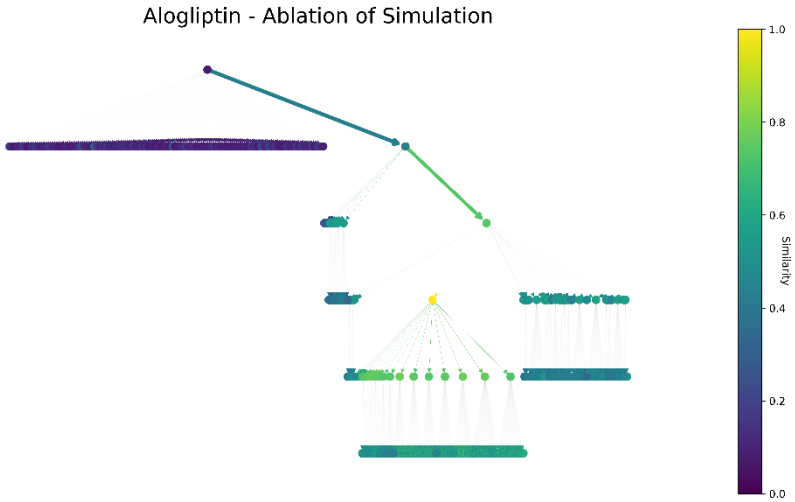


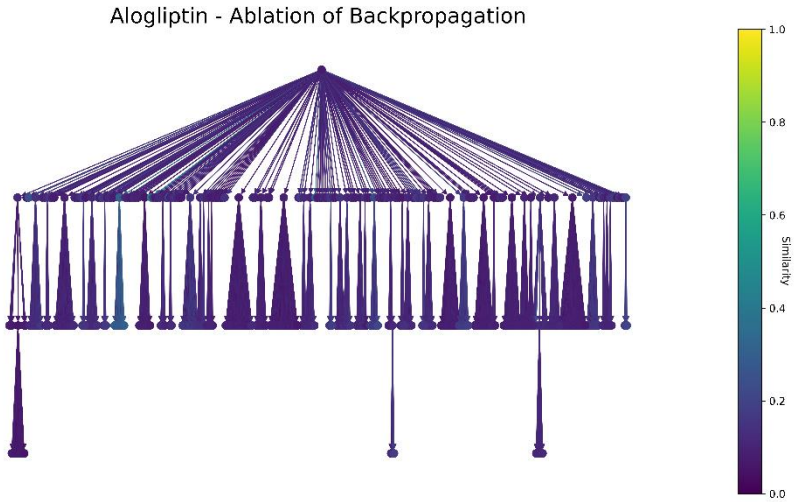



Alogliptin - Ablation of Expansion (75% retained)
1.0
0.8
0.6
Similarity
0.4
0.2
0.0

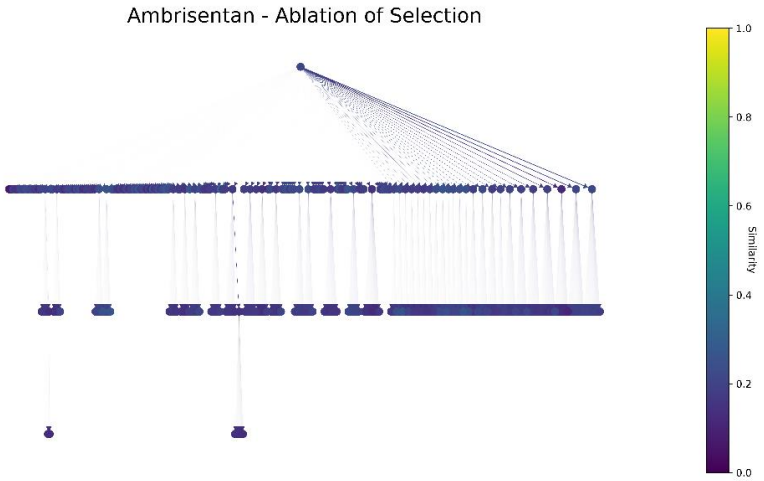

Alogliptin - Ablation of Expansion (50% retained)
1.0
0.8
0.6
Similarity
0.4
0.2
0.0

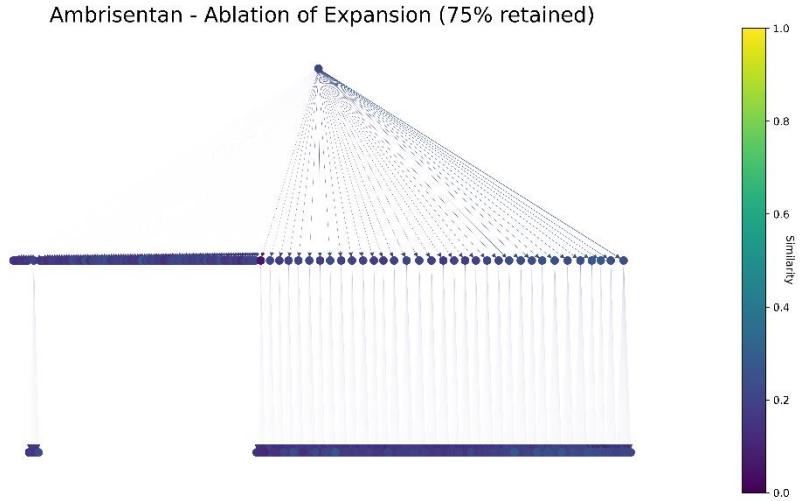

Alogliptin - Ablation of Expansion (25% retained)
1.0
0.8
0.6
Similarity
0.4
0.2
0.0

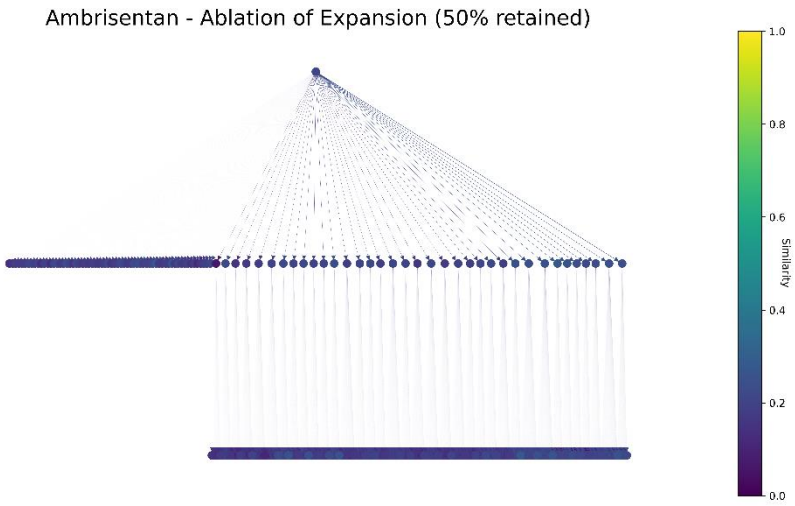

Alogliptin - Ablation of Simulation
1.0
0.8
0.6
Similarity
0.4
0.2
0.0

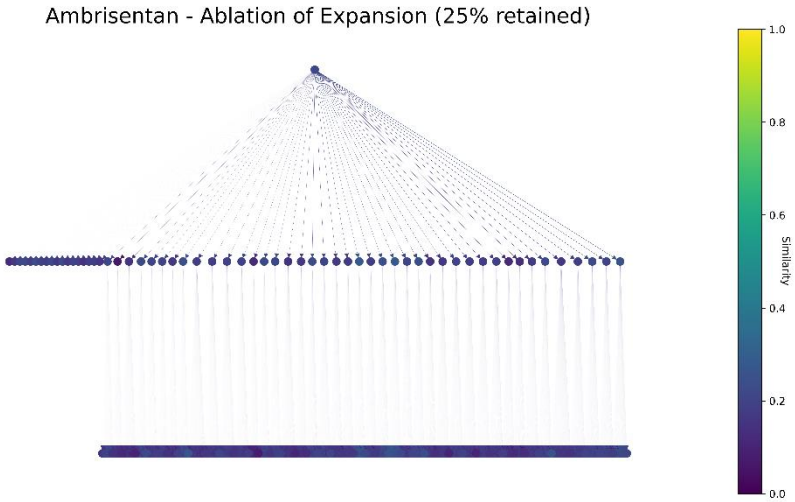


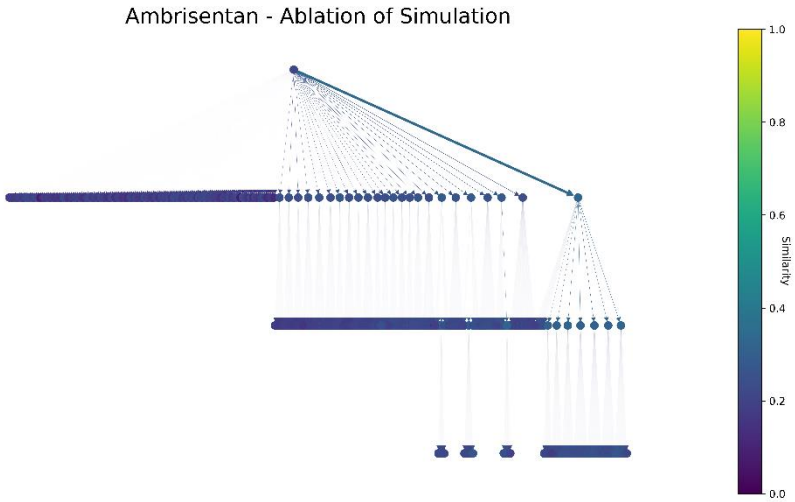
D-2 Selected ambrisentan trees (ablation studies)

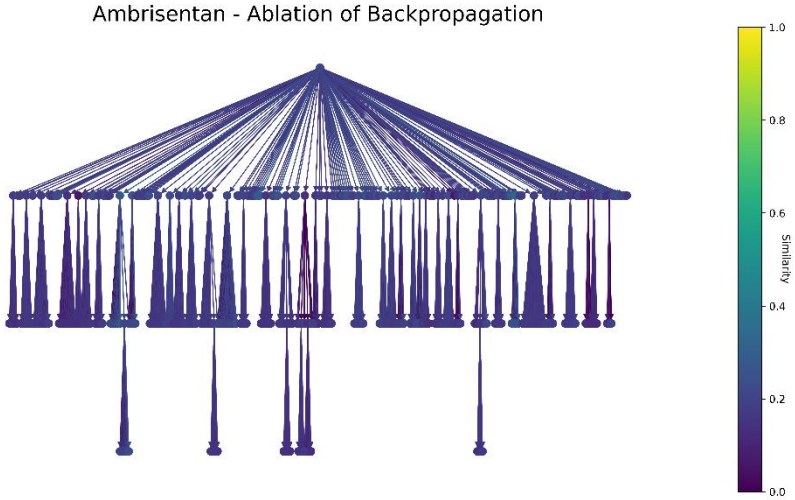

Ambrisentan - Ablation of Selection
1.0
0.8
0.6
Similarity
0.4
0.2
0.0


Ambrisentan - Ablation of Expansion (75% retained)
1.0
0.8
0.6
Similarity
0.4
0.2
0.0


Ambrisentan - Ablation of Expansion (50% retained)
1.0
0.8
0.6
Similarity
0.4
0.2
0.0

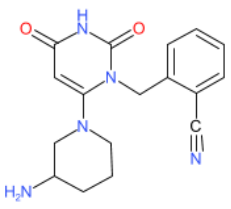
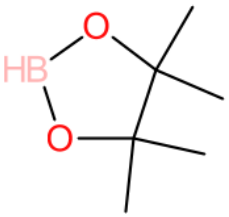
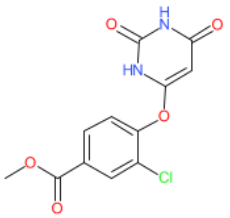
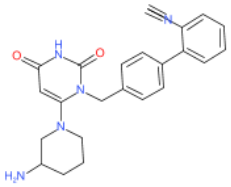
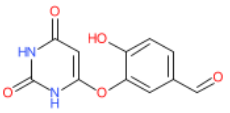
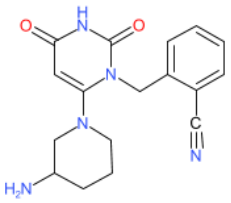
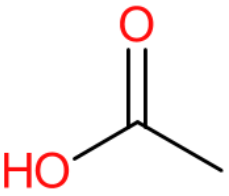
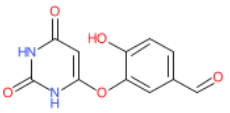
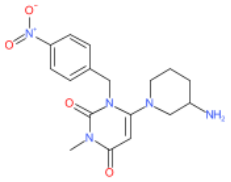
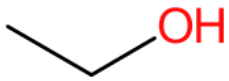
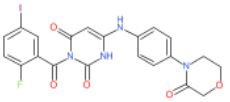
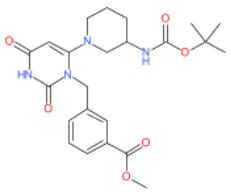
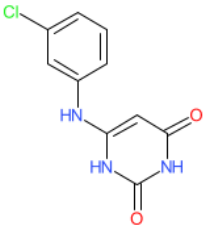
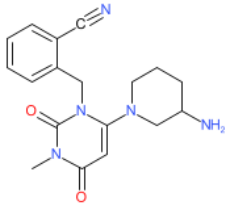
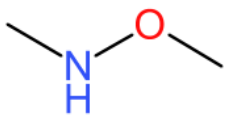
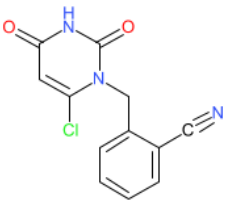

Ambrisentan - Ablation of Expansion (25% retained)
1.0
0.8
0.6
Similarity
0.4
0.2
0.0

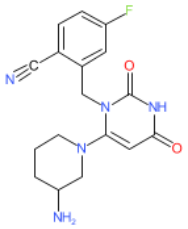
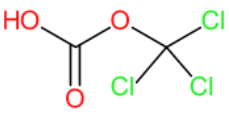

Ambrisentan - Ablation of Simulation
1.0
0.8
0.6
Similarity
0.4
0.2
0.0


Ambrisentan - Ablation of Backpropagation
1.0
0.8
0.6
Similarity
0.4
0.2
0.0

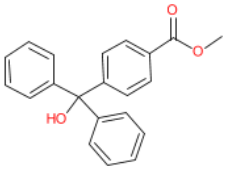
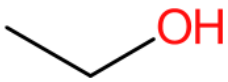
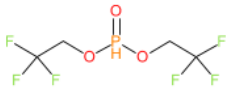
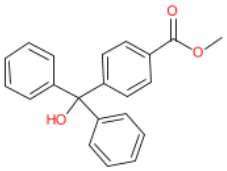
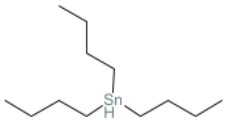
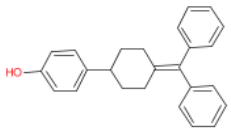
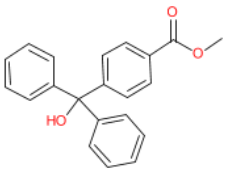
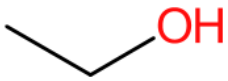
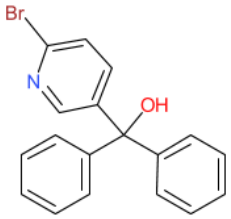
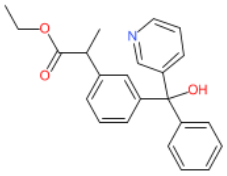
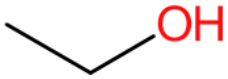
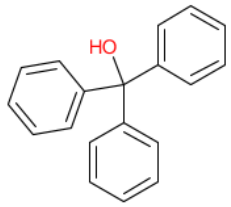
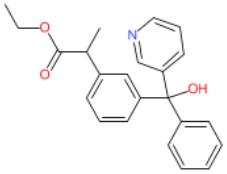
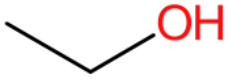
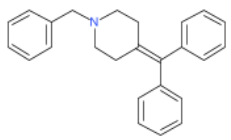
Appendix E Ablation studies selected molecules

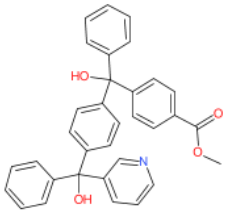
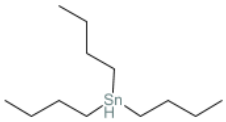
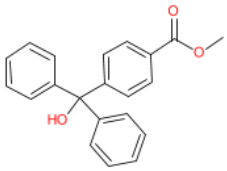
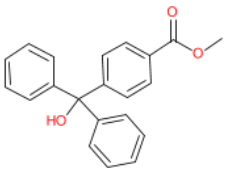
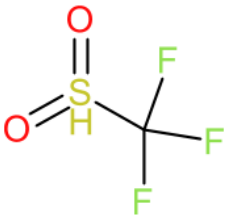
E-1 Alogliptin ablation studies selected molecules

Experiment	Maximum	Minimum	Minimum most visited
Baseline			
Ablation of Selection			
Ablation of Expansion (75% retained)			
Ablation of Expansion (50% retained)			
Ablation of Expansion (25% retained)			
Ablation of Simulation			

Ablation of Backpropagation			Cannot be calculated due to no visit counts being available.

E-2 Ambrisentan Ablation studies selected molecules

Experiment	Maximum	Minimum	Minimum most visited
Baseline			
Ablation of Selection			
Ablation of Expansion (75% retained)			
Ablation of Expansion (50% retained)			
Ablation of Expansion (25% retained)			

Ablation of Simulation			
Ablation of Backpropagation			Cannot be calculated due to no visit counts being available.

Appendix F Calculated values for simulation investigations

F-1 Simulations (random) calculated Max Sim (Tree) (Mean), Max Sim (Tree) (Standard deviation), Max Sim (Tree+Sim) (Mean), Max Sim (Tree+Sim) (Standard deviation)

Experiment	Name	Max Sim (Tree) (Mean)	Max Sim (Tree) (Standard deviation)	Max Sim (Tree+Sim) (Mean)	Max Sim (Tree+Sim) (Standard deviation)
Simulations 1	Aripiprazole	0.799	0.047	0.650	0.039
Simulations 50	Aripiprazole	0.803	0.000	0.812	0.015
Simulations 100	Aripiprazole	0.812	0.012	0.809	0.040
Simulations 1	Ambrisentan	0.403	0.058	0.401	0.099
Simulations 50	Ambrisentan	0.380	0.061	0.578	0.030
Simulations 100	Ambrisentan	0.333	0.000	0.610	0.047

F-2 Simulations (random) calculated Average (Tree) (Mean), Average (Tree) (Standard deviation), Skewness (Mean), Skewness (Standard deviation)

Experiment	Name	Average (Tree) (Mean)	Average (Tree) (Standard deviation)	Skewness (Mean)	Skewness (Standard deviation)
Simulations 1	Aripiprazole	0.248	0.025	0.655	0.777
Simulations 50	Aripiprazole	0.311	0.019	0.644	0.391
Simulations 100	Aripiprazole	0.290	0.013	0.774	0.172
Simulations 1	Ambrisentan	0.186	0.003	0.337	0.092
Simulations 50	Ambrisentan	0.177	0.002	0.499	0.108
Simulations 100	Ambrisentan	0.178	0.003	0.469	0.048

F-3 Simulations (random) calculated Depth (Mean), Depth (Standard deviation), Branching Factor (Mean), Branching Factor (Standard deviation)

Experiment	Name	Depth (Mean)	Depth (Standard deviation)	Branching Factor (Mean)	Branching Factor (Standard deviation)
Simulations 1	Aripiprazole	3.000	0.000	216.630	36.940
Simulations 50	Aripiprazole	3.000	0.000	302.986	28.810
Simulations 100	Aripiprazole	3.000	0.000	268.595	28.652
Simulations 1	Ambrisentan	2.000	0.000	207.325	14.205
Simulations 50	Ambrisentan	2.000	0.000	211.039	8.695
Simulations 100	Ambrisentan	2.000	0.000	199.949	8.827

F-4 Simulations (random) calculated Time (Mean), Time (Standard deviation), Size (Tree) (Mean), Size (Tree) (Standard deviation)

Experiment	Name	Time (Mean)	Time (Standard deviation)	Size (Tree) (Mean)	Size (Tree) (Standard deviation)
Simulations 1	Aripiprazole	13.905	2.399	8463.800	1589.974
Simulations 50	Aripiprazole	110.874	30.598	1766.800	396.036
Simulations 100	Aripiprazole	416.202	142.981	3244.000	1097.814
Simulations 1	Ambrisentan	31.675	2.707	10573.600	724.443
Simulations 50	Ambrisentan	1346.769	85.862	10763.000	443.464
Simulations 100	Ambrisentan	2510.961	174.578	10197.400	450.157

F-5 Simulations(ML) calculated Max Sim (Tree) (Mean), Max Sim (Tree) (Standard deviation), Average (Tree) (Mean), Average (Tree) (Standard deviation)

Experiment	Name	Max Sim (Tree) (Mean)	Max Sim (Tree) (Standard deviation)	Average (Tree) (Mean)	Average (Tree) (Standard deviation)
ML Simulation GMean	Aripiprazole	0.723	0.000	0.337	0.000
ML Simulation Max	Aripiprazole	0.723	0.000	0.169	0.000
ML Simulation Mean	Aripiprazole	0.825	0.000	0.343	0.000
ML Simulation GMean	Ambrisentan	0.333	0.000	0.212	0.000
ML Simulation Max	Ambrisentan	0.340	0.000	0.204	0.000
ML Simulation Mean	Ambrisentan	0.340	0.000	0.213	0.000

F-6 Simulations(ML) calculated Skewness (Mean), Skewness (Standard deviation), Depth (Mean), Depth (Standard deviation)

Experiment	Name	Skewness (Mean)	Skewness (Standard deviation)	Depth (Mean)	Depth (Standard deviation)
ML Simulation GMean	Aripiprazole	-0.542	0.000	3.000	0.000
ML Simulation Max	Aripiprazole	1.124	0.000	3.000	0.000
ML Simulation Mean	Aripiprazole	-0.781	0.002	3.000	0.000
ML Simulation GMean	Ambrisentan	-1.850	0.000	3.000	0.000
ML Simulation Max	Ambrisentan	-1.463	0.000	3.000	0.000
ML Simulation Mean	Ambrisentan	-1.959	0.000	4.000	0.000

F-7 Simulations(ML) calculated Branching Factor (Mean), Branching Factor (Standard deviation), Time (Mean), Time (Standard deviation)

Experiment	Name	Branching Factor (Mean)	Branching Factor (Standard deviation)	Time (Mean)	Time (Standard deviation)
ML Simulation GMean	Aripiprazole	324.892	0.000	3.473	0.071
ML Simulation Max	Aripiprazole	619.588	0.000	8.681	0.109
ML Simulation Mean	Aripiprazole	129.435	0.717	1.673	0.055
ML Simulation GMean	Ambrisentan	204.510	0.000	3.318	0.079
ML Simulation Max	Ambrisentan	315.216	0.000	4.788	0.073
ML Simulation Mean	Ambrisentan	229.490	0.000	3.588	0.095

Appendix G Calculated values for modification investigations

G-1 Modifications calculated Max Sim (Tree) (Mean), Max Sim (Tree) (Standard deviation), Average (Tree) (Mean), Average (Tree) (Standard deviation)

Experiment	Name	Max Sim (Tree) (Mean)	Max Sim (Tree) (Standard deviation)	Average (Tree) (Mean)	Average (Tree) (Standard deviation)
Linear-Max SAUCT	Aripiprazole	0.723	0.000	0.240	0.000
Geom-Max SAUCT	Aripiprazole	0.723	0.000	0.240	0.000
PUCT	Aripiprazole	0.723	0.000	0.208	0.000
PhysChem-Filter	Aripiprazole	0.723	0.000	0.352	0.000
LCS-Filter	Aripiprazole	0.719	0.000	0.340	0.001
RVPrune-Pick1	Aripiprazole	0.356	0.000	0.270	0.000
RVPrune-Pick5	Aripiprazole	1.000	0.000	0.406	0.001
RVPrune-Pick10	Aripiprazole	0.803	0.000	0.388	0.000
RVPrune-Pick15	Aripiprazole	0.803	0.000	0.368	0.000
RVPrune-Pick20	Aripiprazole	0.803	0.000	0.368	0.000
Linear-Max SAUCT	Ambrisentan	1.000	0.000	0.210	0.000
Geom-Max SAUCT	Ambrisentan	1.000	0.000	0.210	0.000
PUCT	Ambrisentan	0.447	0.000	0.195	0.000
PhysChem-Filter	Ambrisentan	0.316	0.000	0.197	0.000
LCS-Filter	Ambrisentan	0.352	0.000	0.259	0.000
RVPrune-Pick1	Ambrisentan	0.333	0.000	0.260	0.000
RVPrune-Pick5	Ambrisentan	0.365	0.000	0.253	0.000
RVPrune-Pick10	Ambrisentan	0.353	0.000	0.250	0.001
RVPrune-Pick15	Ambrisentan	0.353	0.000	0.247	0.000
RVPrune-Pick20	Ambrisentan	0.351	0.000	0.242	0.000

G-2 Modifications calculated Skewness (Mean), Skewness (Standard deviation), Depth (Mean), Depth (Standard deviation)

Experiment	Name	Skewness (Mean)	Skewness (Standard deviation)	Depth (Mean)	Depth (Standard deviation)
Linear-Max SAUCT	Aripiprazole	1.494	0.000	3.000	0.000
Geom-Max SAUCT	Aripiprazole	1.494	0.000	3.000	0.000
PUCT	Aripiprazole	0.822	0.000	2.000	0.000
PhysChem-Filter	Aripiprazole	-0.716	0.000	3.000	0.000
LCS-Filter	Aripiprazole	0.130	0.009	3.000	0.000
RVPrune-Pick1	Aripiprazole			1.000	0.000
RVPrune-Pick5	Aripiprazole	0.432	0.044	3.000	0.000
RVPrune-Pick10	Aripiprazole	0.231	0.000	3.000	0.000

RVPrune-Pick15	Aripiprazole	0.273	0.000	3.000	0.000
RVPrune-Pick20	Aripiprazole	0.012	0.000	3.000	0.000
Linear-Max SAUCT	Ambrisentan	2.947	0.109	4.000	0.000
Geom-Max SAUCT	Ambrisentan	2.908	0.109	4.000	0.000
PUCT	Ambrisentan	-0.907	0.000	2.000	0.000
PhysChem-Filter	Ambrisentan	0.114	0.013	3.000	0.000
LCS-Filter	Ambrisentan	-1.222	0.026	4.000	0.000
RVPrune-Pick1	Ambrisentan	0.186	0.000	4.000	0.000
RVPrune-Pick5	Ambrisentan	-2.012	0.027	4.000	0.000
RVPrune-Pick10	Ambrisentan	-2.192	0.199	4.000	0.000
RVPrune-Pick15	Ambrisentan	-1.495	0.005	4.000	0.000
RVPrune-Pick20	Ambrisentan	-1.751	0.039	3.000	0.000

G-3 Modifications calculated Branching Factor (Mean), Branching Factor (Standard deviation), Time (Mean), Time (Standard deviation)

Experiment	Name	Branching Factor (Mean)	Branching Factor (Standard deviation)	Time (Mean)	Time (Standard deviation)
Linear-Max SAUCT	Aripiprazole	0.240	0.000	2.510	0.146
Geom-Max SAUCT	Aripiprazole	0.240	0.000	2.593	0.218
PUCT	Aripiprazole	0.208	0.000	2.636	0.149
PhysChem-Filter	Aripiprazole	0.352	0.000	2.215	0.118
LCS-Filter	Aripiprazole	0.340	0.001	0.848	0.070
RVPrune-Pick1	Aripiprazole	0.270	0.000	0.003	0.005
RVPrune-Pick5	Aripiprazole	0.406	0.001	0.328	0.109
RVPrune-Pick10	Aripiprazole	0.388	0.000	0.261	0.112
RVPrune-Pick15	Aripiprazole	0.368	0.000	0.334	0.103
RVPrune-Pick20	Aripiprazole	0.368	0.000	0.278	0.092
Linear-Max SAUCT	Ambrisentan	0.210	0.000	2.234	0.225
Geom-Max SAUCT	Ambrisentan	0.210	0.000	2.067	0.154
PUCT	Ambrisentan	0.195	0.000	2.542	0.267
PhysChem-Filter	Ambrisentan	0.197	0.000	3.447	0.219
LCS-Filter	Ambrisentan	0.259	0.000	3.155	0.251
RVPrune-Pick1	Ambrisentan	0.260	0.000	0.092	0.042
RVPrune-Pick5	Ambrisentan	0.253	0.000	0.273	0.064
RVPrune-Pick10	Ambrisentan	0.250	0.001	0.510	0.140
RVPrune-Pick15	Ambrisentan	0.247	0.000	0.717	0.128
RVPrune-Pick20	Ambrisentan	0.242	0.000	0.972	0.166

Appendix H Cross validation performance for simulation models

H-1 CNS CV Values

Problem	Morgan Fingerprint	CV value
CNS	1024 Bits Radius=2	0.637±0.003
CNS	1024 Bits Radius=3	0.615±0.005
CNS	2048 Bits Radius=2	0.676±0.003
CNS	2048 Bits Radius=3	0.654±0.003
CNS	4096 Bits Radius=2	0.699±0.002
CNS	4096 Bits Radius=3	0.673±0.003
CNS	8192 Bits Radius=2	0.712±0.002
CNS	8192 Bits Radius=3	0.681±0.003
CNS	16384 Bits Radius=2	0.726±0.003
CNS	16384 Bits Radius=3	0.681±0.004

H-2 QED CV Values

Problem	Morgan Fingerprint	CV value
QED	1024 Bits Radius=2	0.647±0.002
QED	1024 Bits Radius=3	0.629±0.002
QED	2048 Bits Radius=2	0.692±0.003
QED	2048 Bits Radius=3	0.67±0.002
QED	4096 Bits Radius=2	0.722±0.002
QED	4096 Bits Radius=3	0.697±0.002
QED	8192 Bits Radius=2	0.738±0.001
QED	8192 Bits Radius=3	0.71±0.004
QED	16384 Bits Radius=2	0.752±0.002
QED	16384 Bits Radius=3	0.723±0.003

H-3 Scaffold Hopping CV Values

Problem	Morgan Fingerprint	CV value
Scaffold Hopping	1024 Bits Radius=2	0.44±0.01
Scaffold Hopping	1024 Bits Radius=3	0.403±0.029
Scaffold Hopping	2048 Bits Radius=2	0.518±0.006
Scaffold Hopping	2048 Bits Radius=3	0.474±0.021
Scaffold Hopping	4096 Bits Radius=2	0.585±0.013

Scaffold Hopping	4096 Bits Radius=3	0.554±0.008
Scaffold Hopping	8192 Bits Radius=2	0.617±0.013
Scaffold Hopping	8192 Bits Radius=3	0.62±0.006
Scaffold Hopping	16384 Bits Radius=2	0.625±0.012
Scaffold Hopping	16384 Bits Radius=3	0.63±0.007

H-4 Alectinib CV Values

Problem	**Morgan Fingerprint**	**CV value**
Alectinib	1024 Bits Radius=2	0.699±0.005
Alectinib	1024 Bits Radius=3	0.661±0.007
Alectinib	2048 Bits Radius=2	0.739±0.004
Alectinib	2048 Bits Radius=3	0.68±0.008
Alectinib	4096 Bits Radius=2	0.777±0.004
Alectinib	4096 Bits Radius=3	0.731±0.003
Alectinib	8192 Bits Radius=2	0.775±0.004
Alectinib	8192 Bits Radius=3	0.724±0.007
Alectinib	16384 Bits Radius=2	0.744±0.004
Alectinib	16384 Bits Radius=3	0.616±0.007

H-5 Cariprazine CV Values

Problem	**Morgan Fingerprint**	**CV value**
Cariprazine	1024 Bits Radius=2	0.763±0.005
Cariprazine	1024 Bits Radius=3	0.691±0.006
Cariprazine	2048 Bits Radius=2	0.798±0.003
Cariprazine	2048 Bits Radius=3	0.733±0.005
Cariprazine	4096 Bits Radius=2	0.823±0.003
Cariprazine	4096 Bits Radius=3	0.775±0.005
Cariprazine	8192 Bits Radius=2	0.815±0.003
Cariprazine	8192 Bits Radius=3	0.76±0.007
Cariprazine	16384 Bits Radius=2	0.778±0.004
Cariprazine	16384 Bits Radius=3	0.643±0.008

H-6 Osimertinib CV Values

Problem	**Morgan Fingerprint**	**CV value**
Osimertinib	1024 Bits Radius=2	0.7±0.007
Osimertinib	1024 Bits Radius=3	0.625±0.008
Osimertinib	2048 Bits Radius=2	0.738±0.005
Osimertinib	2048 Bits Radius=3	0.67±0.007
Osimertinib	4096 Bits Radius=2	0.76±0.004
Osimertinib	4096 Bits Radius=3	0.708±0.005

Osimertinib	8192 Bits Radius=2	0.761±0.004
Osimertinib	8192 Bits Radius=3	0.708±0.007
Osimertinib	16384 Bits Radius=2	0.734±0.005
Osimertinib	16384 Bits Radius=3	0.615±0.01

H-7 Pimavanserin CV Values

Problem	**Morgan Fingerprint**	**CV value**
Pimavanserin	1024 Bits Radius=2	0.738±0.005
Pimavanserin	1024 Bits Radius=3	0.658±0.006
Pimavanserin	2048 Bits Radius=2	0.792±0.003
Pimavanserin	2048 Bits Radius=3	0.716±0.008
Pimavanserin	4096 Bits Radius=2	0.819±0.003
Pimavanserin	4096 Bits Radius=3	0.771±0.003
Pimavanserin	8192 Bits Radius=2	0.813±0.003
Pimavanserin	8192 Bits Radius=3	0.761±0.006
Pimavanserin	16384 Bits Radius=2	0.774±0.004
Pimavanserin	16384 Bits Radius=3	0.651±0.008

Appendix I Summary statistics of synthetic tractability scores for REACTS Searches

I-1 CNS MPO summary synthetic tractability

Statistic	**SAScore**	**SCScore**	**RAScore**
mean	2.688	3.492	0.934
std	0.885	0.763	0.146
min	1.070	1.079	0.048
25%	2.073	2.972	0.969
50%	2.473	3.587	0.991
75%	2.933	4.077	0.995
max	5.668	5.000	0.999

I-2 QED summary synthetic tractability

Statistic	***SAScore***	***SCScore***	***RAScore***
mean	2.822	3.885	0.955

std	0.933	0.862	0.111
min	1.070	1.079	0.063
25%	2.198	3.272	0.978
50%	2.529	4.090	0.992
75%	3.139	4.590	0.996
max	5.229	5.000	0.999

I-3 Scaffold hopping summary synthetic tractability

Statistic	SAScore	SCScore	RAScore
mean	2.687	3.943	0.911
std	0.458	0.645	0.159
min	1.383	1.079	0.037
25%	2.320	3.549	0.912
50%	2.689	4.005	0.978
75%	3.040	4.424	0.992
max	4.178	5.000	0.999

I-4 Alectinib summary synthetic tractability

Statistic	SAScore_Alectinib	SCScore_Alectinib	RAScore_Alectinib
mean	2.421	3.931	0.972
std	0.344	0.701	0.064
min	1.383	1.079	0.214
25%	2.226	3.545	0.980
50%	2.408	4.039	0.991
75%	2.593	4.454	0.995
max	4.968	5.000	0.999

I-5 Cariprazine summary synthetic tractability

Statistic	SAScore_Cariprazine	SCScore_Cariprazine	RAScore_Cariprazine
mean	2.324	4.090	0.989
std	0.311	0.730	0.031
min	1.360	1.079	0.286
25%	2.120	3.626	0.992
50%	2.324	4.199	0.995
75%	2.519	4.694	0.997
max	4.888	5.000	1.000

I-6 Osimertinib summary synthetic tractability

Statistic	SAScore_Osimertinib	SCScore_Osimertinib	RAScore_Osimertinib
mean	2.380	4.088	0.981

std	0.379	0.626	0.055
min	1.325	1.079	0.221
25%	2.144	3.735	0.986
50%	2.369	4.187	0.994
75%	2.579	4.564	0.996
max	4.832	5.000	0.999

I-7 Pimavanserin summary synthetic tractability

Statistic	**SAScore_Pimavanserin**	**SCScore_Pimavanserin**	**RAScore_Pimavanserin**
mean	2.347	4.054	0.984
std	0.350	0.665	0.040
min	1.383	1.079	0.209
25%	2.126	3.713	0.988
50%	2.349	4.192	0.994
75%	2.534	4.542	0.997
max	4.968	5.000	0.999

Bibliography

A.T. Balaban. (1967). Chemical graphs. III. Reactions with cyclic six-membered transition states. *Rev. Roum. Chim.*, *12*, 875–898.

Abramson, D., Krishnamoorthy, M., & Dang, H. (1999). Simulated annealing cooling schedules for the school timetabling problem. *Asia-Pacific Journal of Operational Research*, *16*(1), 1–22.

Adamson, G. W., Cowell, J., Lynch, M. F., Mclure, A. H. W., Town, W. G., & Margaret Yapp, A. (1973). Strategic Considerations in the Design of a Screening System for Substructure Searches of Chemical Structure Files. *Journal of Chemical Documentation*, *13*(3), 153–157. https://doi.org/10.1021/c160050a013

Agarwal, A., Chapelle, O., Dudík, M., & Langford, J. (2014). A reliable effective terascale linear learning system. In *Journal of Machine Learning Research* (Vol. 15, pp. 1111–1133).

Agarwal, K. K., Larsen, T. D. L., & Gelernter, H. L. (1978). Application of chemical transforms in synchem2, a computer program for organic synthesis route discovery. *Computers and Chemistry*, *2*(2), 75–84. https://doi.org/10.1016/0097-8485(78)87005-3

Akutsu, T. (2004). Efficient extraction of mapping rules of atoms from enzymatic reaction data. *Journal of Computational Biology*, *11*(2–3), 449–462. https://doi.org/10.1089/1066527041410337

Alexander, D. L. J., Tropsha, A., & Winkler, D. A. (2015). Beware of R2: Simple, Unambiguous Assessment of the Prediction Accuracy of QSAR and QSPR Models. *Journal of Chemical Information and Modeling*, *55*(7), 1316–1322. https://doi.org/10.1021/acs.jcim.5b00206

Alpaydin, E. (2014). *Introduction to Machine Learning* (3rd ed.). MIT Press.

Apostolakis, J., Sacher, O., Körner, R., & Gasteiger, J. (2008). Automatic determination of reaction mappings and reaction center information. 2. Validation on a biochemical reaction database. *Journal of Chemical Information and Modeling*, *48*(6), 1190–1198. https://doi.org/10.1021/ci700433d

Avellaneda, F. (2019). *Learning Optimal Decision Trees from Large Datasets.* https://doi.org/10.48550/arXiv.1904.06314.

Baell, J. B., & Holloway, G. A. (2010). New Substructure Filters for Removal of Pan Assay Interference Compounds (PAINS) from Screening Libraries and for Their Exclusion in Bioassays. *Journal of Medicinal Chemistry*, *53*(7), 2719–2740. https://doi.org/10.1021/JM901137J

Baker, D. B., Horiszny, J. W., & Metanomski, W. V. (1980). History of abstracting at chemical abstracts service. *Journal of Chemical Information and Computer Sciences®*, *20*(4), 193–201. https://doi.org/10.1021/ci60024a001

Baldi, P. (1995). Gradient Descent Learning Algorithm Overview: A General Dynamical Systems Perspective. *IEEE Transactions on Neural Networks*, *6*(1), 182–195. https://doi.org/10.1109/72.363438

Barnard, J. M., & Downs, G. M. (1992). Clustering of chemical structures on the basis of two-dimensional similarity measures. *Journal of Chemical Information and Computer Sciences*, *32*(6), 644–649. https://doi.org/10.1021/ci00010a010

Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G. C., & Stewart, W. R. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, *1*(1), 9–32. https://doi.org/10.1007/BF02430363

Barrett, J. P. (1974). The coefficient of determination-some limitations. *American Statistician*, *28*(1), 19–20. https://doi.org/10.1080/00031305.1974.10479056

Baylon, J. L., Cilfone, N. A., Gulcher, J. R., & Chittenden, T. W. (2019). Enhancing Retrosynthetic Reaction Prediction with Deep Learning Using Multiscale Reaction Classification. *Journal of Chemical Information and Modeling*, *59*(2), 673–688. https://doi.org/10.1021/acs.jcim.8b00801

Beccari, A. R., Cavazzoni, C., Beato, C., & Costantino, G. (2013). LiGen: A high performance workflow for chemistry driven de novo design. *Journal of Chemical Information and Modeling*, *53*(6), 1518–1527. https://doi.org/10.1021/ci400078g

Bellman, R. E. (1961). Adaptive Control Processes. In *Adaptive Control Processes*. Princeton University Press. https://doi.org/10.1515/9781400874668

BenevolentAI/guacamol: Benchmarks for generative chemistry. (2021). https://github.com/BenevolentAI/guacamol

Besnard, J., Ruda, G. F., Setola, V., Abecassis, K., Rodriguiz, R. M., Huang, X. P., Norval, S., Sassano, M. F., Shin, A. I., Webster, L. A., Simeons, F. R. C., Stojanovski, L., Prat, A., Seidah, N. G., Constam, D. B., Bickerton, G. R., Read, K. D., Wetsel, W. C., Gilbert, I. H., … Hopkins, A. L. (2012). Automated design of ligands to polypharmacological profiles. *Nature*, *492*(7428), 215–220. https://doi.org/10.1038/nature11691

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., & Hopkins, A. L. (2012). Quantifying the chemical beauty of drugs. *Nature Chemistry 2011 4:2*, *4*(2), 90–98.

https://doi.org/10.1038/nchem.1243

Bietti, A., Agarwal, A., & Langford, J. (2021). A contextual bandit bake-off. *Journal of Machine Learning Research*, *22*.

Birmingham, J., & Kent, P. (1988). Tree-Searching and Tree-Pruning Techniques. In *Computer Chess Compendium* (pp. 123–128). Springer New York. https://doi.org/10.1007/978-1-4757-1968-0_13

Bjornsson, Y., & Finnsson, H. (2009). CadiaPlayer: A simulation-based general game player. *IEEE Transactions on Computational Intelligence and AI in Games*, *1*(1), 4–15. https://doi.org/10.1109/TCIAIG.2009.2018702

Blake, J. E., & Dana, R. C. (1990). CASREACT: More than a Million Reactions. *Journal of Chemical Information and Computer Sciences*, *30*(4), 394–399. https://doi.org/10.1021/ci00068a008

Blum, A., Kalai, A., & Langford, J. (1999). *Beating the hold-out*. 203–208. https://doi.org/10.1145/307400.307439

Blurock, E. S. (1990). Computer-Aided Synthesis Design at RISC-Linz: Automatic Extraction and Use of Reaction Classes. *Journal of Chemical Information and Computer Sciences*, *30*(4), 505–510. https://doi.org/10.1021/ci00068a024

Bohacek, R. S., McMartin, C., & Guida, W. C. (1996). The art and practice of structure-based drug design: A molecular modeling perspective. In *Medicinal Research Reviews* (Vol. 16, Issue 1, pp. 3–50). Wiley Subscription Services, Inc., A Wiley Company. https://doi.org/10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6

Böhm, H. J. (1992a). The computer program LUDI: A new method for the de novo design of enzyme inhibitors. *Journal of Computer-Aided Molecular Design*, *6*(1), 61–78. https://doi.org/10.1007/BF00124387

Böhm, H. J. (1992b). LUDI: rule-based automatic design of new substituents for enzyme inhibitor leads. *Journal of Computer-Aided Molecular Design*, *6*(6), 593–606. https://doi.org/10.1007/BF00126217

Böhm, H. J. (1993). A novel computational tool for automated structure-based drug design. *Journal of Molecular Recognition : JMR*, *6*(3), 131–137. https://doi.org/10.1002/jmr.300060305

Böhm, H. J. (1994). The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure. *Journal of*

Computer-Aided Molecular Design, *8*(3), 243–256. https://doi.org/10.1007/BF00126743

Bosc, N., Atkinson, F., Felix, E., Gaulton, A., Hersey, A., & Leach, A. R. (2019). Large scale comparison of QSAR and conformal prediction methods and their applications in drug discovery. *Journal of Cheminformatics*, *11*(1), 4. https://doi.org/10.1186/s13321-018-0325-4

Boschelli, D. H., Wang, Y. D., Johnson, S., Wu, B., Ye, F., Barrios Sosa, A. C., Golas, J. M., & Boschelli, F. (2004). 7-Alkoxy-4-phenylamino-3-quinolinecarbonitriles as Dual Inhibitors of Src and Abl Kinases. *Journal of Medicinal Chemistry*, *47*(7), 1599–1601. https://doi.org/10.1021/jm0499458

Bottou, L., Curtis, F. E., & Nocedal, J. (2018). *Optimization Methods for Large-Scale Machine Learning.* https://doi.org/10.48550/arXiv.1606.04838.

Bradshaw, J., Paige, B., Kusner, M. J., Segler, M. H. S., & Hernández-Lobato, J. M. (2019). A Model to Search for Synthesizable Molecules. *Advances in Neural Information Processing Systems*, *32*.

Bradshaw, J., Paige, B., Kusner, M. J., Segler, M. H. S., & Hernández-Lobato, J. M. (2020). *Barking up the right tree: an approach to search over molecule synthesis DAGs.* https://doi.org/10.48550/arXiv.2012.11522.

Breiman, L. (2001). Random Forests. *Machine Learning 2001 45:1*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Broughton, H., Hunt, P., & MacKey, M. (2003). *Methods for classifying and searching chemical reactions*. Google Patents.

Brown, N., Fiscato, M., Segler, M. H. S., & Vaucher, A. C. (2019). GuacaMol: Benchmarking Models for de Novo Molecular Design. *Journal of Chemical Information and Modeling*, *59*(3), 1096–1108. https://doi.org/10.1021/acs.jcim.8b00839

Brown, N., McKay, B., Gilardoni, F., & Gasteiger, J. (2004). A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. *Journal of Chemical Information and Computer Sciences*, *44*(3), 1079–1087. https://doi.org/10.1021/ci034290p

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., & Colton, S. (2012). A survey of Monte Carlo tree search methods. In *IEEE Transactions on Computational Intelligence and AI in Games* (Vol. 4, Issue 1, pp. 1–43). https://doi.org/10.1109/TCIAIG.2012.2186810

Brudno, A. (1963). Bounds and valuations for shortening the search of estimates. *Problemy Kibernetiki*, *10*, 141–150.

Budin, N., Majeux, N., Tenette–Souaille, C., & Caflisch, A. (2001). Structure-based ligand design by a build-up approach and genetic algorithm search in conformational space. *Journal of Computational Chemistry*, *22*(16), 1956–1970. https://doi.org/10.1002/JCC.1145

Butina, D. (1999). Unsupervised Data Base Clustering Based on Daylight's Fingerprint and Tanimoto Similarity: A Fast and Automated Way To Cluster Small and Large Data Sets. *Journal of Chemical Information and Computer Sciences*, *39*(4), 747–750. https://doi.org/10.1021/ci9803381

Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O., & Walsh, A. (2018). Machine learning for molecular and materials science. *Nature*, *559*(7715), 547–555. https://doi.org/10.1038/s41586-018-0337-2

Button, A., Merk, D., Hiss, J. A., & Schneider, G. (2019). Automated de novo molecular design by hybrid machine intelligence and rule-driven chemical synthesis. *Nature Machine Intelligence*, *1*(7), 307–315. https://doi.org/10.1038/s42256-019-0067-7

Carhart, R. E., Smith, D. H., & Venkataraghavan, R. (1985). Atom pairs as molecular features in structure-activity studies: definition and applications. *Journal of Chemical Information and Modeling*, *25*(2), 64–73. https://doi.org/10.1021/ci00046a002

Carlson, N. R., & Birkett, M. A. (2017). Physiology of Behavior. In *Physiology of Behavior* (pp. 21–22). Pearson Education, Limited.

CAS Reactions. (2021). https://www.cas.org/cas-data/cas-reactions

Cereto-Massagué, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallvé, S., & Pujadas, G. (2015). Molecular fingerprint similarity search in virtual screening. *Methods*, *71*(C), 58–63. https://doi.org/10.1016/j.ymeth.2014.08.005

Chaslot, G. M. J. B., Winands, M. H., & Van Den Herik, H. J. (2008). Parallel Monte-Carlo tree search. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5131 LNCS*, 60–71. https://doi.org/10.1007/978-3-540-87608-3_6

Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., & Blaschke, T. (2018). The rise of deep learning in drug discovery. *Drug Discovery Today*, *23*(6), 1241–1250. https://doi.org/https://doi.org/10.1016/j.drudis.2018.01.039

Chen, L., & Gasteiger, J. (1997). Knowledge discovery in reaction databases: Landscaping organic reactions by a self-organizing neural network. *Journal of the American Chemical Society*, *119*(17), 4033–4042. https://doi.org/10.1021/ja960027b

Chen, W. L., Chen, D. Z., & Taylor, K. T. (2013). Automatic reaction mapping and reaction center detection. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *3*(6), 560–593. https://doi.org/10.1002/wcms.1140

Chéron, N., Jasty, N., & Shakhnovich, E. I. (2015). OpenGrowth: An Automated and Rational Algorithm for Finding New Protein Ligands. *Journal of Medicinal Chemistry*, *59*(9), 4171–4188. https://doi.org/10.1021/ACS.JMEDCHEM.5B00886

Chollet, F., & others. (2015). *Keras*.

Clark, D. E., Frenkel, D., Levy, S. A., Li, J., Murray, C. W., Robson, B., Waszkowycz, B., & Westhead, D. R. (1995). PRO-LIGAND: an approach to de novo molecular design. 1. Application to the design of organic molecules. *Journal of Computer-Aided Molecular Design*, *9*(1), 13–32.

Coley, C. W., Rogers, L., Green, W. H., & Jensen, K. F. (2018). SCScore: Synthetic Complexity Learned from a Reaction Corpus. *Journal of Chemical Information and Modeling*, *58*(2), 252–261. https://doi.org/10.1021/ACS.JCIM.7B00622

Collette, Y., & Siarry, P. (2004). Multiobjective Optimization. In *Decision Engineering*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-08883-8

Congreve, M., Carr, R., Murray, C., & Jhoti, H. (2003). A 'Rule of Three' for fragment-based lead discovery? *Drug Discovery Today*, *8*(19), 876–877. https://doi.org/10.1016/S1359-6446(03)02831-9

Corey, E. J. (1967). General methods for the construction of complex molecules. *Pure and Applied Chemistry*, *14*(1), 19–38. https://doi.org/10.1351/PAC196714010019

Corey, E. J., Wipke, W. T., Iii, R. D. C., & Howe, W. J. (1972). Organic and Biological Chemistry: Computer-Assisted Synthetic Analysis. Facile Man-Machine Communication of Chemical Structure by Interactive Computer Graphics. *Journal of the American Chemical Society*, *94*(2), 421–430. https://doi.org/10.1021/ja00757a020

Cortes, D. (2018). *Adapting multi-armed bandits policies to contextual bandits scenarios.* https://doi.org/10.48550/arXiv.1811.04383.

Coulom, R. (2006). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In H. J. van den Herik, P. Ciancarini, & H. H. L. M. (Jeroen) Donkers (Eds.), *Computers and Games* (pp. 72–83). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-75538-8_7

Crowe, J. E., Lynch, M. F., & Town, W. G. (1970). Analysis of structural characteristics of chemical

compounds in a large computer-based file. Part I. Non-cyclic fragments. *Journal of the Chemical Society C: Organic*, *7*, 990–996. https://doi.org/10.1039/j39700000990

Dai, H., Tian, Y., Dai, B., Skiena, S., & Song, L. (2018). Syntax-directed variational autoencoder for structured data. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Dalby, A., Nourse, J. G., Hounshell, W. D., Gushurst, A. K. I., Grier, D. L., Leland, B. A., & Laufer, J. (1992). Description of Several Chemical Structure File Formats Used by Computer Programs Developed at Molecular Design Limited. *Journal of Chemical Information and Computer Sciences*, *32*(3), 244–255. https://doi.org/10.1021/ci00007a012

Damewood, J. R., Lemian, C. L., & Masek, B. B. (2010). NovoFLAP: A ligand-based de novo design approach for the generation of medicinally relevant ideas. *Journal of Chemical Information and Modeling*, *50*(7), 1296–1303. https://doi.org/10.1021/ci100080r

Danziger, D. J., & Dean, P. M. (1989a). Automated site-directed drug design: a general algorithm for knowledge acquisition about hydrogen-bonding regions at protein surfaces. *Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character. Royal Society (Great Britain)*, *236*(1283), 101–113. https://doi.org/10.1098/rspb.1989.0015

Danziger, D. J., & Dean, P. M. (1989b). Automated site-directed drug design: the prediction and observation of ligand point positions at hydrogen-bonding regions on protein surfaces. *Proc R Soc Lond B Biol Sci*, *236*(1283), 115–124. https://doi.org/10.1098/rspb.1989.0016

Darwin, C. (1859). On the Origin of Species, 1859. In *On the Origin of Species, 1859*. Routledge. https://doi.org/10.4324/9780203509104

Davies, M., Nowotka, M., Papadatos, G., Dedman, N., Gaulton, A., Atkinson, F., Bellis, L., & Overington, J. P. (2015). ChEMBL web services: Streamlining access to drug discovery data and utilities. *Nucleic Acids Research*, *43*(W1), W612–W620. https://doi.org/10.1093/nar/gkv352

Daylight Chemical Information Systems Inc. (n.d.). *Daylight Theory Manual*. 2011. Retrieved 27 September 2021, from https://www.daylight.com/dayhtml/doc/theory/

De Cao, N., & Kipf, T. (2018). *MolGAN: An implicit generative model for small molecular graphs.* https://doi.org/10.48550/arXiv.1805.11973.

De Luca, A., Horvath, D., Marcou, G., Solov'Ev, V., & Varnek, A. (2012). Mining chemical reactions using neighborhood behavior and condensed graphs of reactions approaches. *Journal of Chemical Information and Modeling*, *52*(9), 2325–2338. https://doi.org/10.1021/ci300149n

Dean, P. M., Firth-Clark, S., Harris, W., Kirton, S. B., & Todorov, N. P. (2006). SkelGen: A general tool for structure-based de novo ligand design. *Expert Opinion on Drug Discovery*, *1*(2), 179–189. https://doi.org/10.1517/17460441.1.2.179

Degen, J., & Rarey, M. (2006). FlexNovo: Structure-Based Searching in Large Fragment Spaces. *ChemMedChem*, *1*(8), 854–868. https://doi.org/10.1002/cmdc.200500102

Devillers, J. (1996). Designing molecules with specific properties from intercommunicating hybrid systems. *Journal of Chemical Information and Computer Sciences*, *36*(6), 1061–1066. https://doi.org/10.1021/ci960022y

DeWitte, R. S., Ishchenko, A. V., & Shakhnovich, E. I. (1997). SMoG: de novo design method based on simple, fast, and accurate free energy estimates. 2. Case studies in molecular design. *Journal of the American Chemical Society*, *119*(20), 4608–4617. https://doi.org/10.1021/ja963689+

DeWitte, R. S., & Shakhnovich, E. I. (1996). SMoG: De novo design method based on simple, fast, and accurate free energy estimates. 1. Methodology and supporting evidence. *Journal of the American Chemical Society*, *118*(47), 11733–11744. https://doi.org/10.1021/ja960751u

Dey, F., & Caflisch, A. (2008). Fragment-based de Novo Ligand design by multiobjective evolutionary optimization. *Journal of Chemical Information and Modeling*, *48*(3), 679–690. https://doi.org/10.1021/ci700424b

Doak, B. C., Zheng, J., Dobritzsch, D., & Kihlberg, J. (2015). How Beyond Rule of 5 Drugs and Clinical Candidates Bind to Their Targets. *Journal of Medicinal Chemistry*, *59*(6), 2312–2327. https://doi.org/10.1021/ACS.JMEDCHEM.5B01286

Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, *344*(2–3), 243–278. https://doi.org/10.1016/J.TCS.2005.05.020

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, *2*, 1470–1477. https://doi.org/10.1109/CEC.1999.782657

Douguet, D., Munier-Lehmann, H., Labesse, G., & Pochet, S. (2005). LEA3D: A computer-aided ligand design for structure-based drug design. *Journal of Medicinal Chemistry*, *48*(7), 2457–2468. https://doi.org/10.1021/jm0492296

Duchi, J., Hazan, E., & Singer, Y. (2010). Adaptive subgradient methods for online learning and stochastic optimization. *COLT 2010 - The 23rd Conference on Learning Theory*, *12*, 257–269.

Durant, J. L., Leland, B. A., Henry, D. R., & Nourse, J. G. (2002). Reoptimization of MDL keys for use in drug discovery. *Journal of Chemical Information and Computer Sciences*, *42*(6), 1273–1280. https://doi.org/10.1021/ci010132r

Durrant, J. D., Amaro, R. E., & McCammon, J. A. (2009). AutoGrow: A novel algorithm for protein inhibitor design. *Chemical Biology and Drug Design*, *73*(2), 168–178. https://doi.org/10.1111/j.1747-0285.2008.00761.x

Durrant, J. D., Lindert, S., & McCammon, J. A. (2013). AutoGrow 3.0: An improved algorithm for chemically tractable, semi-automated protein inhibitor design. *Journal of Molecular Graphics and Modelling*, *44*, 104–112. https://doi.org/10.1016/j.jmgm.2013.05.006

Eisen, M. B., Wiley, D. C., Karplus, M., & Hubbard, R. E. (1994). HOOK: A program for finding novel molecular architectures that satisfy the chemical and steric requirements of a macromolecule binding site. *Proteins: Structure, Function, and Bioinformatics*, *19*(3), 199–221. https://doi.org/10.1002/prot.340190305

Ellis, F. (2002). *Paracetamol* (C. Osborne & M. J. Pack (eds.)).The Royal Society of Chemistry.

Elsevier. (2021). https://www.elsevier.com/solutions/reaxys

Elton, D. C., Boukouvalas, Z., Fuge, M. D., & Chung, P. W. (2019). Deep learning for molecular design - A review of the state of the art. *Molecular Systems Design and Engineering*, *4*(4), 828–849. https://doi.org/10.1039/c9me00039a

Enamine. (2021). *Fragment Libraries - Enamine*. https://enamine.net/compound-libraries/fragment-libraries

Ertl, P. (2003). Cheminformatics analysis of organic substituents: Identification of the most common substituents, calculation of substituent properties, and automatic identification of drug-like bioisosteric groups. *Journal of Chemical Information and Computer Sciences*, *43*(2), 374–380. https://doi.org/10.1021/ci0255782

Ertl, P., & Schuffenhauer, A. (2009). Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, *1*(1), 8. https://doi.org/10.1186/1758-2946-1-8

Fechner, U., & Schneider, G. (2006). Flux (1): A Virtual Synthesis Scheme for Fragment-Based de Novo Design. *Journal of Chemical Information and Modeling*, *46*(2), 699–707. https://doi.org/10.1021/ci0503560

Fechner, U., & Schneider, G. (2007). Flux (2): Comparison of Molecular Mutation and Crossover Operators for Ligand-Based de Novo Design. *Journal of Chemical Information and Modeling*, *47*(2), 656–667. https://doi.org/10.1021/ci6005307

Feldman, A., & Hodes, L. (1975). An Efficient Design for Chemical Structure Searching. I. The Screens. *Journal of Chemical Information and Computer Sciences*, *15*(3), 147–152. https://doi.org/10.1021/ci60003a004

Feng, J., Gwaltney II, S. L., Stafford, J. A., & Zang, Z. (2005). Preparation of pyrimidine-2,4-dione compounds as dipeptidyl peptidase IV inhibitors. In *Jpn. Kokai Tokkyo Koho*. Syrrx, Inc., USA .

Firth, N. C., Atrash, B., Brown, N., & Blagg, J. (2015). MOARF, an Integrated Workflow for Multiobjective Optimization: Implementation, Synthesis, and Biological Evaluation. *Journal of Chemical Information and Modeling*, *55*(6), 1169–1180. https://doi.org/10.1021/acs.jcim.5b00073

Flower, D. R. (1998). On the Properties of Bit String-Based Measures of Chemical Similarity. *Journal of Chemical Information and Computer Sciences*, *38*(3), 379–386. https://doi.org/10.1021/ci970437z

Free, S. M., & Wilson, J. W. (1964). A Mathematical Contribution to Structure-Activity Studies. *Journal of Medicinal Chemistry*, *7*(4), 395–399. https://doi.org/10.1021/jm00334a001

Freeland, R. G., Funk, S. A., O'Korn, L. J., & Wilson, G. A. (1979). The Chemical Abstracts Service Chemical Registry System. II. Augmented connectivity molecular formula. *Journal of Chemical Information and Computer Sciences*, *19*(2), 94–98. https://doi.org/10.1021/ci60018a012

Freund, Y., & Schapire, R. E. (1999). *A Short Introduction to Boosting*.

Fujita, S. (1986). Description of Organic Reactions Based on Imaginary Transition Structures. 1. Introduction of New Concepts. *Journal of Chemical Information and Computer Sciences*, *26*(4), 205–212. https://doi.org/10.1021/ci00052a009

Funatsu, K., Endo, T., Kotera, N., & Sasaki, S. I. (1988). Automatic recognition of reaction site in organic chemical reactions. *Tetrahedron Computer Methodology*, *1*(1), 53–69. https://doi.org/10.1016/0898-5529(88)90008-5

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4). https://doi.org/10.1145/2523813

García-Domenech, R., Gálvez, J., de Julián-Ortiz, J. V., & Pogliani, L. (2008). Some new trends in

chemical graph theory. *Chemical Reviews*, *108*(3), 1127–1169. https://doi.org/10.1021/cr0780006

Gasteiger, J., Hutchings, M. G., Christoph, B., Gann, L., Hiller, C., Löw, P., Marsili, M., Saller, H., & Yuki, K. (1987). *A new treatment of chemical reactivity: Development of EROS, an expert system for reaction prediction and synthesis design*. 19–73. https://doi.org/10.1007/3-540-16904-0_14

Gasteiger, J., Ihlenfeldt, W. D., Fick, R., & Rose, J. R. (2002). Similarity concepts for the planning of organic reactions and syntheses. *Journal of Chemical Information and Computer Sciences*, *32*(6), 700–712. https://doi.org/10.1021/CI00010A018

Gehlhaar, D. K., Moerder, K. E., Zichi, D., Sherman, C. J., Ogden, R. C., & Freer, S. T. (1995). De Novo Design of Enzyme Inhibitors by Monte Carlo Ligand Generation. *Journal of Medicinal Chemistry*, *38*(3), 466–472. https://doi.org/10.1021/jm00003a010

Gelernter, H. L., Sanders, A. F., Larsen, D. L., Agarwal, K. K., Boivie, R. H., Spritzer, G. A., & Searleman, J. E. (1977). Empirical explorations of SYNCHEM. *Science*, *197*(4308), 1041–1049. https://doi.org/10.1126/science.197.4308.1041

Gelernter, H., Rose, J. R., & Chen, C. (2002). Building and refining a knowledge base for synthetic organic chemistry via the methodology of inductive and deductive machine learning. *Journal of Chemical Information and Computer Sciences*, *30*(4), 492–504. https://doi.org/10.1021/CI00068A023

Ghiandoni, G. M. (2019). *Enhancing Reaction-based de novo Design using Machine Learning*. University of Sheffield.

Ghiandoni, G. M., Bodkin, M. J., Chen, B., Hristozov, D., Wallace, J. E. A., Webster, J., & Gillet, V. (2019). Development and Application of a Data-Driven Reaction Classification Model: Comparison of an ELN and the Medicinal Chemistry Literature. *Journal of Chemical Information and Modeling*. https://doi.org/10.1021/acs.jcim.9b00537

Ghiandoni, G. M., Bodkin, M. J., Chen, B., Hristozov, D., Wallace, J. E. A., Webster, J., & Gillet, V. J. (2020). Enhancing reaction-based de novo design using a multi-label reaction class recommender. *Journal of Computer-Aided Molecular Design*, *34*(7), 783–803. https://doi.org/10.1007/s10822-020-00300-6

Gillet, V. J., Johnson, A. P., Mata, P., & Sike, S. (1990). Automated structure design in 3D. *Tetrahedron Computer Methodology*, *3*(6), 681–696. https://doi.org/10.1016/0898-5529(90)90167-7

Gillet, V. J., Johnson, A. P., Mata, P., Sike, S., & Williams, P. (1993). SPROUT: A program for structure generation. *Journal of Computer-Aided Molecular Design*, *7*(2), 127–153. https://doi.org/10.1007/BF00126441

Gillet, V. J., Newell, W., Mata, P., Myatt, G., Sike, S., Zsoldos, Z., & Johnson, A. P. (1994). SPROUT: Recent Developments in the de Novo Design of Molecules. *Journal of Chemical Information and Computer Sciences*, *34*(1), 207–217. https://doi.org/10.1021/ci00017a027

Gillet, V. J., Khatib, W., Willett, P., Fleming, P. J., & Green, D. V. S. (2002). Combinatorial library design using a multiobjective genetic algorithm. *Journal of Chemical Information and Computer Sciences*, *42*(2), 375–385. https://doi.org/10.1021/ci010375j

Gillet, V. J., Bodkin, M. J., & Hristozov, D. (2013). Multiobjective De Novo Design of Synthetically Accessible Compounds. In *De novo Molecular Design* (pp. 267–285). Wiley-VCH Verlag GmbH & Co. KGaA. https://doi.org/10.1002/9783527677016.ch11

Glen, R. C., & Payne, A. W. R. (1995). A genetic algorithm for the automated generation of molecules within constraints. *Journal of Computer-Aided Molecular Design 1995 9:2*, *9*(2), 181–202. https://doi.org/10.1007/BF00124408

Godden, J. W., Xue, L., & Bajorath, J. (2000). Combinatorial Preferences Affect Molecular Similarity/Diversity Calculations Using Binary Fingerprints and Tanimoto Coefficients. *Journal of Chemical Information and Computer Sciences*, *40*(1), 163–166. https://doi.org/10.1021/ci990316u

Gohlke, H., Hendlich, M., & Klebe, G. (2000). Knowledge-based scoring function to predict protein-ligand interactions. *Journal of Molecular Biology*, *295*(2), 337–356. https://doi.org/10.1006/jmbi.1999.3371

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., & Aspuru-Guzik, A. (2018). Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, *4*(2), 268–276. https://doi.org/10.1021/acscentsci.7b00572

Gong, Z., Zhong, P., & Hu, W. (2019). Diversity in Machine Learning. *IEEE Access*, *7*, 64323–64350. https://doi.org/10.1109/ACCESS.2019.2917620

GoogleResearch. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*.

Gottipati, S. K., Sattarov, B., Niu, S., Pathak, Y., Wei, H., Liu, S., Thomas, K. J., Blackburn, S., Coley, C. W., Tang, J., Chandar, S., & Bengio, Y. (2020). Learning to navigate the synthetically accessible

chemical space using reinforcement learning. In *37th International Conference on Machine Learning, ICML 2020* (Vol. PartF16814, pp. 3626–3637). https://doi.org/10.48550/arXiv.2004.12485.

Grethe, G., Blanke, G., Kraut, H., & Goodman, J. M. (2018). International chemical identifier for reactions (RInChI). *Journal of Cheminformatics*, *10*(1), 1–9. https://doi.org/10.1186/s13321-018-0277-8

Guedes, I. A., Pereira, F. S. S., & Dardenne, L. E. (2018). Empirical scoring functions for structure-based virtual screening: Applications, critical aspects, and challenges. *Frontiers in Pharmacology*, *9*, 1089. https://doi.org/10.3389/fphar.2018.01089

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., & Aspuru-Guzik, A. (2017). *Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models.* https://doi.org/10.48550/arXiv.1705.10843.

Haque, I. S., Pande, V. S., & Walters, W. P. (2011). Anatomy of High-Performance 2D Similarity Calculations. *Journal of Chemical Information and Modeling*, *51*(9), 2345–2351. https://doi.org/10.1021/ci200235e

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

Hartenfeller, M., Proschak, E., Schüller, A., & Schneider, G. (2008). Concept of combinatorial de novo design of drug-like molecules by particle swarm optimization. *Chemical Biology and Drug Design*, *72*(1), 16–26. https://doi.org/10.1111/j.1747-0285.2008.00672.x

Hartenfeller, M., & Schneider, G. (2011a). De novo drug design. In *Methods in molecular biology (Clifton, N.J.)* (Vol. 672, pp. 299–323). Humana Press, Totowa, NJ. https://doi.org/10.1007/978-1-60761-839-3_12

Hartenfeller, M., & Schneider, G. (2011b). Enabling future drug discovery by de novo design. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *1*(5), 742–759. https://doi.org/10.1002/wcms.49

Hartenfeller, M., Zettl, H., Walter, M., Rupp, M., Reisen, F., Proschak, E., Weggen, S., Stark, H., & Schneider, G. (2012). Dogs: Reaction-driven de novo design of bioactive compounds. *PLoS Computational Biology*, *8*(2), e1002380. https://doi.org/10.1371/journal.pcbi.1002380

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer New

York Inc.

Hawkins, P. C. D. (2017). Conformation Generation: The State of the Art. *Journal of Chemical Information and Modeling*, *57*(8), 1747–1756. https://doi.org/10.1021/acs.jcim.7b00221

Heller, S. R., McNaught, A., Pletnev, I., Stein, S., & Tchekhovskoi, D. (2015). InChI, the IUPAC International Chemical Identifier. *Journal of Cheminformatics*, *7*(1), 1–34. https://doi.org/10.1186/s13321-015-0068-4

Hendrickson, J. B. (1997). Comprehensive System for Classification and Nomenclature of Organic Reactions. *Journal of Chemical Information and Computer Sciences*, *37*(5), 852–860. https://doi.org/10.1021/CI970040V

Hiss, J. A., Reutlinger, M., Koch, C. P., Perna, A. M., Schneider, P., Rodrigues, T., Haller, S., Folkers, G., Weber, L., Baleeiro, R. B., Walden, P., Wrede, P., & Schneider, G. (2014). Combinatorial chemistry by ant colony optimization. *Future Medicinal Chemistry*, *6*(3), 267–280. https://doi.org/10.4155/fmc.13.203

Ho, C. M. W., & Marshall, G. R. (1993). SPLICE: A program to assemble partial query solutions from three-dimensional database searches into novel ligands. *Journal of Computer-Aided Molecular Design*, *7*(6), 623–647. https://doi.org/10.1007/BF00125322

Hodes, L. (1976). Selection of Descriptors According to Discrimination and Redundancy. Application to Chemical Structure Searching. *Journal of Chemical Information and Computer Sciences*, *16*(2), 88–93. https://doi.org/10.1021/ci60006a012

Hoksza, D., Škoda, P., Voršilák, M., & Svozil, D. (2014). Molpher: a software framework for systematic chemical space exploration. *Journal of Cheminformatics 2014 6:1*, *6*(1), 1–13. https://doi.org/10.1186/1758-2946-6-7

Holliday, J. D., Salim, N., Whittle, M., & Willett, P. (2003). Analysis and Display of the Size Dependence of Chemical Similarity Coefficients. *Journal of Chemical Information and Computer Sciences*, *43*(3), 819–828. https://doi.org/10.1021/ci034001x

Horwood, J., & Noutahi, E. (2020). Molecular Design in Synthetically Accessible Chemical Space via Deep Reinforcement Learning. *ACS Omega*, *5*(51), 32984–32994. https://doi.org/10.1021/ACSOMEGA.0C04153

Hristozov, D., Bodkin, M., Chen, B., Patel, H., & Gillet, V. J. (2011). Validation of reaction vectors for de novo design. *ACS Symposium Series*, *1076*, 29–43. https://doi.org/10.1021/bk-2011-1076.ch002

Huang, Q., Li, L.-L., & Yang, S.-Y. (2010). PhDD: A new pharmacophore-based de novo design method of drug-like molecules combined with assessment of synthetic accessibility. *Journal of Molecular Graphics and Modelling*, *28*(8), 775–787. https://doi.org/10.1016/j.jmgm.2010.02.002

InfoChem - DeepMatter. (2021). https://www.deepmatter.io/about-us/infochem

Irwin, J. J., Tang, K. G., Young, J., Dandarchuluun, C., Wong, B. R., Khurelbaatar, M., Moroz, Y. S., Mayfield, J., & Sayle, R. A. (2020). ZINC20 - A Free Ultralarge-Scale Chemical Database for Ligand Discovery. *Journal of Chemical Information and Modeling*, *60*(12), 6065–6073. https://doi.org/10.1021/acs.jcim.0c00675

Ishchenko, A. V., & Shakhnovich, E. I. (2002). SMall Molecule Growth 2001 (SMoG2001): An improved knowledge-based scoring function for protein-ligand interactions. *Journal of Medicinal Chemistry*, *45*(13), 2770–2780. https://doi.org/10.1021/jm0105833

James, G., Daniela, W., Trevor, H., & Robert, T. (2021). *An introduction to statistical learning: with applications in R* (Second). Spinger.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, *20*(4), 422–446. https://doi.org/10.1145/582415.582418

Jaworski, W., Szymkuć, S., Mikulak-Klucznik, B., Piecuch, K., Klucznik, T., Kaźmierowski, M., Rydzewski, J., Gambin, A., & Grzybowski, B. A. (2019). Automatic mapping of atoms across both simple and complex chemical reactions. *Nature Communications*, *10*(1). https://doi.org/10.1038/s41467-019-09440-2

Jensen, J. H. (2019a). A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. *Chemical Science*, *10*(12), 3567–3572. https://doi.org/10.1039/c8sc05372c

Johnson, M. A., & Maggiora, G. M. (1990). *Concepts and applications of molecular similarity*. Wiley.

Kelder, J., Grootenhuis, P. D. J., Bayada, D. M., Delbressine, L. P. C., & Ploemen, J.-P. (1999). Polar Molecular Surface as a Dominating Determinant for Oral Absorption and Brain Penetration of Drugs. *Pharmaceutical Research 1999 16:10*, *16*(10), 1514–1519. https://doi.org/10.1023/A:1015040217741

Kiefer, J., & Wolfowitz, J. (1952). Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, *23*(3), 462–466. https://doi.org/10.1214/aoms/1177729392

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Knuth, D. E., & Moore, R. W. (1975). An analysis of alpha-beta pruning. *Artificial Intelligence*, *6*(4), 293–326.

Kocsis, L., & Szepesvári, C. (2006). Bandit Based Monte-Carlo Planning. In J. Fürnkranz, T. Scheffer, & M. Spiliopoulou (Eds.), *Machine Learning: ECML 2006* (pp. 282–293). Springer Berlin Heidelberg. https://doi.org/10.1007/11871842_29

Kocsis, L., Szepesvári, C., & Willemson, J. (2006). Improved Monte-Carlo Search. *White Paper*, *1*, 22. https://doi.org/10.1007/11871842_29

Korf, R. E. (1985). Depth-first iterative-deepening. An optimal admissible tree search. *Artificial Intelligence*, *27*(1), 97–109. https://doi.org/10.1016/0004-3702(85)90084-0

Korovina, K., Xu, S., Kandasamy, K., Neiswanger, W., Poczos, B., Schneider, J., & Xing, E. P. (2019). *ChemBO: Bayesian Optimization of Small Organic Molecules with Synthesizable Recommendations.* https://doi.org/10.48550/arXiv.1908.01425.

Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, *4*(2), 87–112. https://doi.org/10.1007/BF00175355

Kraut, H., Eiblmaier, J., Grethe, G., Löw, P., Matuszczyk, H., & Saller, H. (2013). Algorithm for reaction classification. *Journal of Chemical Information and Modeling*, *53*(11), 2884–2895. https://doi.org/10.1021/ci400442f

Kumar, Akhil, & Maranas, C. D. (2014). CLCA: Maximum common molecular substructure queries within the MetRxn database. *Journal of Chemical Information and Modeling*, *54*(12), 3417–3438. https://doi.org/10.1021/ci5003922

Kumar, A. V, Rameshchandra, U. A., Mansukhlal, T. N., Patel, B., & Kantibhai, P. (2016). Process for preparing clobazam using novel intermediates. In *PCT Int. Appl.* Amneal Pharmaceuticals Company GmbH, Switz. .

Kutchukian, P. S., Virtanen, S. I., Lounkine, E., Glick, M., & Shakhnovich, E. I. (2013). Construction of drug-like compounds by markov chains. In *De novo Molecular Design* (pp. 311–323). Wiley-VCH Verlag GmbH & Co. KGaA. https://doi.org/10.1002/9783527677016.ch13

Kwon, Y., & Lee, J. (2021). MolFinder: an evolutionary algorithm for the global optimization of molecular properties and the extensive exploration of chemical space using SMILES. *Journal of*

Cheminformatics, *13*(1), 1–14. https://doi.org/10.1186/s13321-021-00501-7

Lajiness, M. (1991). Evaluation of the performance of dissimilarity selection methodology. In S. C. & V. A. (Eds.), *QSAR: Rational approaches to the design of bioactive compounds* (pp. 201–204).

Lameijer, E. W., Kok, J. N., Bäck, T., & Ijzerman, A. P. (2006). The molecule evoluator. An interactive evolutionary algorithm for the design of drug-like molecules. *Journal of Chemical Information and Modeling*, *46*(2), 545–552. https://doi.org/10.1021/ci050369d

Langford, J., & Zhang, T. (2009). The Epoch-Greedy algorithm for contextual multi-armed bandits. *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*.

Lee, C. Y. (1961). An Algorithm for Path Connections and Its Applications. *IRE Transactions on Electronic Computers*, *EC-10*(3), 346–365. https://doi.org/10.1109/TEC.1961.5219222

Lenselink, E. B., ten Dijke, N., Bongers, B., Papadatos, G., van Vlijmen, H. W. T., Kowalczyk, W., IJzerman, A. P., & van Westen, G. J. P. (2017). Beyond the hype: deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set. *Journal of Cheminformatics*, *9*(1), 45. https://doi.org/10.1186/s13321-017-0232-0

Lewell, X. Q., Judd, D. B., Watson, S. P., & Hann, M. M. (1998). RECAP - Retrosynthetic Combinatorial Analysis Procedure: A powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry. *Journal of Chemical Information and Computer Sciences*, *38*(3), 511–522. https://doi.org/10.1021/ci970429i

Lewis, R A, & Dean, P. M. (1989a). Automated site-directed drug design: the concept of spacer skeletons for primary structure generation. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, *236*(1283), 125–140. https://doi.org/10.1098/RSPB.1989.0017

Lewis, R A, & Dean, P. M. (1989b). Automated site-directed drug design: the formation of molecular templates in primary structure generation. *Proceedings of the Royal Society of London. Series B, Containing Papers of a Biological Character. Royal Society (Great Britain)*, *236*(1283), 141–162. https://doi.org/10.1098/rspb.1989.0018

Lewis, Richard A. (1990). Automated site-directed drug design: Approaches to the formation of 3D molecular graphs. *Journal of Computer-Aided Molecular Design*, *4*(2), 205–210. https://doi.org/10.1007/BF00125319

Lewis, Richard A., Roe, D. C., Huang, C., Ferrin, T. E., Langridge, R., & Kuntz, I. D. (1992). Automated site-directed drug design using molecular lattices. *Journal of Molecular Graphics*, *10*(2), 66–78. https://doi.org/10.1016/0263-7855(92)80059-M

Lewis, Richard A., & Wood, D. (2014). Modern 2D QSAR for drug discovery. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *4*(6), 505–522. https://doi.org/10.1002/wcms.1187

Lipinski, C. A., Lombardo, F., Dominy, B. W., & Feeney, P. J. (1997). Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. In *Advanced Drug Delivery Reviews* (Vol. 23, Issues 1–3, pp. 3–25). Elsevier Science B.V. https://doi.org/10.1016/S0169-409X(96)00423-1

Litsa, E. E., Pena, M. I., Moll, M., Giannakopoulos, G., Bennett, G. N., & Kavraki, L. E. (2019). Machine Learning Guided Atom Mapping of Metabolic Reactions. *Journal of Chemical Information and Modeling*, *59*(3), 1121–1135. https://doi.org/10.1021/acs.jcim.8b00434

Liu, T., Naderi, M., Alvin, C., Mukhopadhyay, S., & Brylinski, M. (2017). Break Down in Order to Build Up: Decomposing Small Molecules for Fragment-Based Drug Design with eMolFrag. *Journal of Chemical Information and Modeling*, *57*(4), 627–631. https://doi.org/10.1021/acs.jcim.6b00596

Liu, X., IJzerman, A. P., & van Westen, G. J. P. (2021). Computational Approaches for De Novo Drug Design: Past, Present, and Future. In *Methods in Molecular Biology* (Vol. 2190, pp. 139–165). Humana, New York, NY. https://doi.org/10.1007/978-1-0716-0826-5_6

Liu, X., Ye, K., van Vlijmen, H. W. T., IJzerman, A. P., & van Westen, G. J. P. (2019). An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: A case for the adenosine A2A receptor. *Journal of Cheminformatics*, *11*(1), 1–16. https://doi.org/10.1186/s13321-019-0355-6

Lo, Y. C., Rensi, S. E., Torng, W., & Altman, R. B. (2018). Machine learning in chemoinformatics and drug discovery. *Drug Discovery Today*, *23*(8), 1538–1546. https://doi.org/10.1016/J.DRUDIS.2018.05.010

Lowe, D. (2017). Chemical reactions from US patents (1976-Sep2016). *URL Https://Figshare. Com/Articles/Chemical_ Reactions_from_US_patents_1976-Sep2016_/5104873*. https://doi.org/10.6084/m9.figshare.5104873.v1

Lowe, D. M. (2012). Extraction of chemical structures and reactions from the literature. In *Doctoral Thesis* (Issue June). https://doi.org/10.17863/CAM.16293

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

Luo, Z., Wang, R., & Lai, L. (1996). RASSE: A New Method for Structure-Based Drug Design. *Journal of*

Chemical Information and Modeling, *36*(6), 1187–1194. https://doi.org/10.1021/ci950277w

Ma, B., Terayama, K., Matsumoto, S., Isaka, Y., Sasakura, Y., Iwata, H., Araki, M., & Okuno, Y. (2021). Structure-Based de Novo Molecular Generator Combined with Artificial Intelligence and Docking Simulations. *Journal of Chemical Information and Modeling*, *61*(7), 3304–3313. https://doi.org/10.1021/ACS.JCIM.1C00679

Maddison, C. J., Huang, A., Sutskever, I., & Silver, D. (2015). Move evaluation in go using deep convolutional neural networks. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings.* https://doi.org/10.48550/arXiv.1412.6564.

Maggiora, G., Vogt, M., Stumpfe, D., & Bajorath, J. (2013). Molecular Similarity in Medicinal Chemistry. *Journal of Medicinal Chemistry*, *57*(8), 3186–3204. https://doi.org/10.1021/JM401411Z

Maragakis, P., Nisonoff, H., Cole, B., & Shaw, D. E. (2020). A deep-learning view of chemical space designed to facilitate drug discovery. *Journal of Chemical Information and Modeling*, *60*(10), 4487–4496. https://doi.org/10.1021/acs.jcim.0c00321

Mata, P., Gillet, V. J., Peter Johnson, A., Lampreia, J., Myatt, G. J., Sike, S., & Stebbings, A. L. (1995). SPROUT: 3D Structure Generation Using Templates. *Journal of Chemical Information and Computer Sciences*, *35*(3), 479–493. https://doi.org/10.1021/ci00025a016

McDonagh, J. L., Van Mourik, T., & Mitchell, J. B. O. (2015). Predicting Melting Points of Organic Molecules: Applications to Aqueous Solubility Prediction Using the General Solubility Equation. *Molecular Informatics*, *34*(11–12), 715–724. https://doi.org/10.1002/minf.201500052

McGrath, N. A., Brichacek, M., & Njardarson, J. T. (2010). A Graphical Journey of Innovative Organic Architectures That Have Improved Our Lives. *Journal of Chemical Education*, *87*(12), 1348–1349. https://doi.org/10.1021/ed1003806

Mcgregor, J. J., & Willett, P. (1981). Use of a Maximal Common Subgraph Algorithm in the Automatic Identification of the Ostensible Bond Changes Occurring in Chemical Reactions. *Journal of Chemical Information and Computer Sciences*, *21*(3), 137–140. https://doi.org/10.1021/ci00031a005

McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, *3*(29), 861. https://doi.org/10.21105/joss.00861

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M. P., Mosquera, J. F., Mutowo, P., Nowotka, M., Gordillo-Marañón, M., Hunter, F., Junco, L.,

Mugumbate, G., Rodriguez-Lopez, M., Atkinson, F., Bosc, N., Radoux, C. J., Segura-Cabrera, A., … Leach, A. R. (2019). ChEMBL: Towards direct deposition of bioassay data. *Nucleic Acids Research*, *47*(D1), D930–D940. https://doi.org/10.1093/nar/gky1075

Meng, X.-Y., Zhang, H.-X., Mezei, M., & Cui, M. (2012). Molecular Docking: A Powerful Approach for Structure-Based Drug Discovery. *Current Computer Aided-Drug Design*, *7*(2), 146–157. https://doi.org/10.2174/157340911795677602

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

Meyers, J., Fabian, B., & Brown, N. (2021). De novo molecular design and generative models. *Drug Discovery Today*. https://doi.org/10.1016/j.drudis.2021.05.019

Michalewicz, Z. (1996). Genetic algorithms + data structures = evolution programs (3rd, extended ed.). In *Genetic algorithms + data structures = evolution programs (3rd, extended ed.)*. Springer-Verlag New York, Inc.

Miller, J., & Ting, T. (2019). EoN (Epidemics on Networks): a fast, flexible Python package for simulation, analytic approximation, and analysis of epidemics on networks. *Journal of Open Source Software*, *4*(44), 1731. https://doi.org/10.21105/joss.01731

Miranker, A., & Karplus, M. (1991). Functionality maps of binding sites: A multiple copy simultaneous search method. *Proteins: Structure, Function, and Bioinformatics*, *11*(1), 29–34. https://doi.org/10.1002/prot.340110104

Moore, E. F. (1959). The shortest path through a maze. *Int. Symposium on the Theory of Switching, Part II*, 285–292.

Morgan, H. L. (1965). The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *J Chem Docum*, *5*. https://doi.org/10.1021/c160017a018

Muegge, I. (2006). PMF scoring revisited. *Journal of Medicinal Chemistry*, *49*(20), 5895–5902. https://doi.org/10.1021/jm050038s

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Murray-Rust, P. (2008). Open Data in Science. *Nature Precedings*, 1–1. https://doi.org/10.1038/npre.2008.1526.1

Murray, C. W., Clark, D. E., Auton, T. R., Firth, M. A., Li, J., Sykes, R. A., Waszkowycz, B., Westhead, D. R., & Young, S. C. (1997). PRO_SELECT: Combining structure-based drug design and combinatorial chemistry for rapid lead discovery. 1. Technology. *Journal of Computer-Aided Molecular Design 1997 11:2*, *11*(2), 193–207. https://doi.org/10.1023/A:1008094712424

Murray, C. W., & Rees, D. C. (2009). The rise of fragment-based drug discovery. *Nature Chemistry*, *1*(3), 187–192. https://doi.org/10.1038/nchem.217

Nachbar, R. B. (1998). Molecular Evolution: A Hierarchical Representation for Chemical Topology and Its Automated Manipulation. *Genetic Programming 1998*, 246–253.

Nachbar, R. B. (2000). Molecular Evolution: Automated Manipulation of Hierarchical Chemical Topology and Its Application to Average Molecular Structures. *Genetic Programming and Evolvable Machines*, *1*(1/2), 57–94. https://doi.org/10.1023/A:1010072431120

Nasu, Y. (2018). *Efficiently Updatable Neural-Network-based Evaluation Functions for Computer Shogi*.

Neudert, G., & Klebe, G. (2011). DSX: A Knowledge-Based Scoring Function for the Assessment of Protein–Ligand Complexes. *Journal of Chemical Information and Modeling*, *51*(10), 2731–2745. https://doi.org/10.1021/CI200274Q

Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, 615–622. https://doi.org/10.1145/1015330.1015435

Nicolaou, C. A., & Brown, N. (2013). Multi-objective optimization methods in drug design. In *Drug Discovery Today: Technologies* (Vol. 10, Issue 3, pp. e427–e435). Elsevier. https://doi.org/10.1016/j.ddtec.2013.02.001

Nicolaou, C. A., Brown, N., & Pattichis, C. S. (2007). Molecular optimization using computational multi-objective methods. *Current Opinion in Drug Discovery & Development*, *10 3*, 316–324.

Nigam, A., Friederich, P., Krenn, M., & Aspuru-Guzik, A. (2019). *Augmenting Genetic Algorithms with Deep Neural Networks for Exploring the Chemical Space.* https://doi.org/10.48550/arXiv.1909.11655.

Nigam, A., Pollice, R., Krenn, M., Gomes, G. dos P., & Aspuru-Guzik, A. (2021). Beyond generative models: superfast traversal, optimization, novelty, exploration and discovery (STONED) algorithm for molecules using SELFIES. *Chemical Science*, *12*(20), 7079–7090. https://doi.org/10.1039/D1SC00231G

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using maximum entropy for text classification. *IJCAI-99 Workshop on Machine Learning for Information Filtering*.

Nishibata, Y., & Itai, A. (1991). Automatic creation of drug candidate structures based on receptor structure. Starting point for artificial lead generation. *Tetrahedron*, *47*(43), 8985–8990. https://doi.org/10.1016/S0040-4020(01)86503-0

Nishibata, Y., & Itai, A. (1993). Confirmation of Usefulness of a Structure Construction Program Based on Three-Dimensional Receptor Structure for Rational Lead Generation. *Journal of Medicinal Chemistry*, *36*(20), 2921–2928. https://doi.org/10.1021/jm00072a011

Nisius, B., & Rester, U. (2009). Fragment Shuffling: An Automated Workflow for Three-Dimensional Fragment-Based Ligand Design. *Journal of Chemical Information and Modeling*, *49*(5), 1211–1222. https://doi.org/10.1021/ci8004572

Nugmanov, R. I., Mukhametgaleev, R. N., Akhmetshin, T., Gimadiev, T. R., Afonina, V. A., Madzhidov, T. I., & Varnek, A. (2019). CGRtools: Python Library for Molecule, Reaction, and Condensed Graph of Reaction Processing. *Journal of Chemical Information and Modeling*, *59*(6), 2516–2521. https://doi.org/10.1021/ACS.JCIM.9B00102

Olivecrona, M., Blaschke, T., Engkvist, O., & Chen, H. (2017). Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, *9*(1), 48. https://doi.org/10.1186/s13321-017-0235-x

Open Reaction Database. (2021). https://docs.open-reaction-database.org/en/latest/overview.html

Openmolecules. (2021). https://openmolecules.org/webreactions/index.html

Organic Syntheses. (2021). http://www.orgsyn.org

Oshiro, Y., Sato, S., & Kurahashi, N. (1989). *US5006528: Carbostyril derivatives*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. D. Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 32, pp. 8024–8035). Curran Associates, Inc.

Patel, H. (2009). *Knowledge-Based De Novo Design using Reaction Vectors*. University of Sheffield.

Patel, H., Bodkin, M. J., Chen, B., & Gillet, V. J. (2009). Knowledge-based approach to de Novo design

using reaction vectors. *Journal of Chemical Information and Modeling*, *49*(5), 1163–1184. https://doi.org/10.1021/ci800413m

Pearlman, D. A., & Murcko, M. A. (1993). CONCEPTS: New dynamic algorithm forde novo drug suggestion. *Journal of Computational Chemistry*, *14*(10), 1184–1193. https://doi.org/10.1002/jcc.540141008

Pearlman, D. A., & Murcko, M. A. (1996). CONCERTS: Dynamic Connection of Fragments as an Approach to de Novo Ligand Design. *Journal of Medicinal Chemistry*, *39*(8), 1651–1663. https://doi.org/10.1021/jm950792l

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space . *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

Pegg, S. C., Haresco, J. J., & Kuntz, I. D. (2001). A genetic algorithm for structure-based de novo design. *Journal of Computer-Aided Molecular Design*, *15*(10), 911–933.

Pierce, A. C., Rao, G., & Bemis, G. W. (2004). BREED: Generating novel inhibitors through hybridization of known ligands. Application to CDK2, P38, and HIV protease. *Journal of Medicinal Chemistry*, *47*(11), 2768–2775. https://doi.org/10.1021/jm030543u

Pinto, D. J. P., Orwat, M. J., Koch, S., Rossi, K. A., Alexander, R. S., Smallwood, A., Wong, P. C., Rendina, A. R., Luettgen, J. M., Knabb, R. M., He, K., Xin, B., Wexler, R. R., & Lam, P. Y. S. (2007). Discovery of 1-(4-methoxyphenyl)-7-oxo-6-(4-(2-oxopiperidin-1-yl)phenyl)-4, 5,6,7-tetrahydro-1H-pyrazolo[3,4-c]pyridine-3-carboxamide (Apixaban, BMS-562247), a highly potent, selective, efficacious, and orally bioavailable inhibitor of blood coagulation f. *Journal of Medicinal Chemistry*, *50*(22), 5339–5356. https://doi.org/10.1021/jm070245n

Polishchuk, P. (2020). CReM: Chemically reasonable mutations framework for structure generation. *Journal of Cheminformatics*, *12*(1), 1–18. https://doi.org/10.1186/s13321-020-00431-w

Polishchuk, P. G., Madzhidov, T. I., & Varnek, A. (2013). Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of Computer-Aided Molecular Design*, *27*(8), 675–679. https://doi.org/10.1007/s10822-013-9672-4

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., Kadurin, A., Johansson, S., Chen, H., Nikolenko, S., Aspuru-Guzik, A., & Zhavoronkov, A. (2020). Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Frontiers in Pharmacology*, *0*, 1931.

https://doi.org/10.3389/FPHAR.2020.565644

Popova, M., Isayev, O., & Tropsha, A. (2018). Deep reinforcement learning for de novo drug design. *Science Advances*, *4*(7). https://doi.org/10.1126/SCIADV.AAP7885

Proschak, E., Sander, K., Zettl, H., Tanrikulu, Y., Rau, O., Schneider, P., Schubert-Zsilavecz, M., Stark, H., & Schneider, G. (2009). From molecular shape to potent bioactive agents II: Fragment-based de novo design. *ChemMedChem*, *4*(1), 45–48. https://doi.org/10.1002/cmdc.200800314

PubChem Substructure Fingerprint v1.3. (2018). ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem_fingerprints.txt

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/bf00116251

Rahman, S. A., Torrance, G., Baldacci, L., Cuesta, S. M., Fenninger, F., Gopal, N., Choudhary, S., May, J. W., Holliday, G. L., Steinbeck, C., & Thornton, J. M. (2016). Reaction Decoder Tool (RDT): Extracting features from chemical reactions. *Bioinformatics*, *32*(13), 2065–2066. https://doi.org/10.1093/bioinformatics/btw096

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. *ACM International Conference Proceeding Series*, *382*, 873–880. https://doi.org/10.1145/1553374.1553486

Ray, L. C., & Kirsch, R. A. (1957). Finding chemical records by digital computers. *Science*, *126*(3278), 814–819. https://doi.org/10.1126/science.126.3278.814

RDKit. (2021). https://rdkit.org/

Reutlinger, M., Rodrigues, T., Schneider, P., & Schneider, G. (2014). Multi-objective molecular de novo design by adaptive fragment prioritization. *Angewandte Chemie - International Edition*, *53*(16), 4244–4248. https://doi.org/10.1002/anie.201310864

Revill, P., Serradell, N., & Bolós, J. (2006). Eltrombopag. Antithrombocytopenic, thrombopoietin receptor agonist. *Drugs of the Future*, *31*(9), 767–770. https://doi.org/10.1358/dof.2006.031.09.1030939

Riechers, H., Albrecht, H. P., Amberg, W., Baumann, E., Bernard, H., Böhm, H. J., Klinge, D., Kling, A., Müller, S., Raschack, M., Unger, L., Walker, N., & Wernet, W. (1996). Discovery and optimization of a novel class of orally active nonpeptidic endothelin-A receptor antagonists. *Journal of Medicinal Chemistry*, *39*(11), 2123–2128. https://doi.org/10.1021/jm960274q

Roatsch, M., Hoffmann, I., Abboud, M. I., Hancock, R. L., Tarhonskaya, H., Hsu, K. F., Wilkins, S. E., Yeh, T. L., Lippl, K., Serrer, K., Moneke, I., Ahrens, T. D., Robaa, D., Wenzler, S., Barthes, N. P. F., Franz, H., Sippl, W., Lassmann, S., Diederichs, S., … Jung, M. (2019). The Clinically Used Iron Chelator Deferasirox Is an Inhibitor of Epigenetic JumonjiC Domain-Containing Histone Demethylases. *ACS Chemical Biology*, *14*(8), 1737–1750. https://doi.org/10.1021/acschembio.9b00289

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586

Robinson, R. L. M., Palczewska, A., Palczewski, J., & Kidley, N. (2017). Comparison of the Predictive Performance and Interpretability of Random Forest and Linear Models on Benchmark Data Sets. *Journal of Chemical Information and Modeling*, *57*(8), 1773–1792. https://doi.org/10.1021/ACS.JCIM.6B00753

Roe, D. C., & Kuntz, I. D. (1995). BUILDER v.2: Improving the chemistry of a de novo design strategy. *Journal of Computer-Aided Molecular Design*, *9*(3), 269–282. https://doi.org/10.1007/BF00124457

Rogers, D., & Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, *50*(5), 742–754. https://doi.org/10.1021/ci100050t

Rose, J. R., & Gasteiger, J. (1994). HORACE: An Automatic System for the Hierarchical Classification of Chemical Reactions. *Journal of Chemical Information and Computer Sciences*, *34*(1), 74–90. https://doi.org/10.1021/ci00017a010

Rosin, C. D. (2011). Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, *61*(3), 203–230. https://doi.org/10.1007/s10472-011-9258-6

Ross, S., Mineiro, P., & Langford, J. (2013). Normalized online learning. *Uncertainty in Artificial Intelligence - Proceedings of the 29th Conference, UAI 2013*, 537–545. https://arxiv.org/abs/1305.6646.

Rotstein, S. H., & Murcko, M. A. (1993a). GenStar: A method for de novo drug design. *Journal of Computer-Aided Molecular Design*, *7*(1), 23–43. https://doi.org/10.1007/BF00141573

Rotstein, S. H., & Murcko, M. A. (1993b). GroupBuild: a fragment-based method for de novo drug design. *Journal of Medicinal Chemistry*, *36*(12), 1700–1710. https://doi.org/10.1021/jm00064a003

rsc-ontologies/rxno: Name Reaction Ontology. (n.d.). Retrieved 22 September 2021, from

https://github.com/rsc-ontologies/rxno

Ruder, S. (2017a). *An overview of gradient descent optimization algorithms.* https://doi.org/10.48550/arXiv.1609.04747.

Ruder, S. (2017b). *An Overview of Multi-Task Learning in Deep Neural Networks.* https://doi.org/10.48550/arXiv.1706.05098.

Ruijl, B., Vermaseren, J., Plaat, A., & Van Den Herik, J. (2014). Combining simulated annealing and Monte Carlo tree search for expression simplification. *Belgian/Netherlands Artificial Intelligence Conference*, 171–172. https://doi.org/10.48550/arXiv.1312.0841.

Rupakheti, C., Virshup, A., Yang, W., & Beratan, D. N. (2015). Strategy to discover diverse optimal molecules in the small molecule universe. *Journal of Chemical Information and Modeling*, *55*(3), 529–537. https://doi.org/10.1021/ci500749q

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM J. Res. Dev.*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Schlimmer, J. C., & Granger, R. H. (1986). Incremental learning from noisy data. *Machine Learning*, *1*(3), 317–354. https://doi.org/10.1007/bf00116895

Schneider, G., & Fechner, U. (2005b). Computer-based de novo design of drug-like molecules. In *Nature Reviews Drug Discovery* (Vol. 4, Issue 8, pp. 649–663). Nature Publishing Group. https://doi.org/10.1038/nrd1799

Schneider, G., Lee, M.-L. L., Stahl, M., & Schneider, P. (2000). De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. *Journal of Computer-Aided Molecular Design*, *14*(5), 487–494. https://doi.org/10.1023/A:1008184403558

Schneider, N., Lowe, D. M., Sayle, R. A., & Landrum, G. A. (2015). Development of a novel fingerprint for chemical reactions and its application to large-scale reaction classification and similarity. *Journal of Chemical Information and Modeling*, *55*(1), 39–53. https://doi.org/10.1021/ci5006614

Schneider, P., Walters, W. P., Plowright, A. T., Sieroka, N., Listgarten, J., Goodnow, R. A., Fisher, J., Jansen, J. M., Duca, J. S., Rush, T. S., Zentgraf, M., Hill, J. E., Krutoholow, E., Kohler, M., Blaney, J., Funatsu, K., Luebkemann, C., & Schneider, G. (2020). Rethinking drug design in the artificial intelligence era. In *Nature Reviews Drug Discovery* (Vol. 19, Issue 5, pp. 353–364). Nature Research. https://doi.org/10.1038/s41573-019-0050-3

Schreck, J. S., Coley, C. W., & Bishop, K. J. M. (2019). Learning Retrosynthetic Planning through Simulated Experience. *ACS Central Science*, *5*(6), 970–981. https://doi.org/10.1021/acscentsci.9b00055

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., & Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, *588*(7839), 604–609. https://doi.org/10.1038/s41586-020-03051-4

Schwaller, P., Laino, T., Gaudin, T., Bolgar, P., Hunter, C. A., Bekas, C., & Lee, A. A. (2019). Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction. *ACS Central Science*, *5*(9), 1572–1583. https://doi.org/10.1021/acscentsci.9b00576

Schwaller, P., Probst, D., Vaucher, A. C., Nair, V. H., Kreutter, D., Laino, T., & Reymond, J.-L. (2021). Mapping the space of chemical reactions using attention-based neural networks. *Nature Machine Intelligence 2021 3:2*, *3*(2), 144–152. https://doi.org/10.1038/s42256-020-00284-w

Scikit-learn. (2021). *3.1. Cross-validation*. https://scikit-learn.org/stable/modules/cross_validation.html

Segler, M. H. S., Kogej, T., Tyrchan, C., & Waller, M. P. (2017). *Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks.* https://doi.org/10.48550/arXiv.1701.01329.

Segler, M. H. S., Preuss, M., & Waller, M. P. (2018a). Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, *555*(7698), 604–610. https://doi.org/10.1038/nature25978

Senger, M. R., Fraga, C. A. M., Dantas, R. F., & Silva, F. P. (2016). Filtering promiscuous compounds in early drug discovery: is it a good idea? *Drug Discovery Today*, *21*(6), 868–872. https://doi.org/10.1016/J.DRUDIS.2016.02.004

Sheridan, R. P., Wang, W. M., Liaw, A., Ma, J., & Gifford, E. M. (2016). Extreme Gradient Boosting as a Method for Quantitative Structure-Activity Relationships. *Journal of Chemical Information and Modeling*, *56*(12), 2353–2360. https://doi.org/10.1021/acs.jcim.6b00591

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. https://doi.org/10.1038/nature16961

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140–1144. https://doi.org/10.1126/science.aar6404

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359. https://doi.org/10.1038/nature24270

Simonovsky, M., & Komodakis, N. (2018). GraphVAE: Towards generation of small graphs using variational autoencoders. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11139 LNCS*, 412–422. https://doi.org/10.1007/978-3-030-01418-6_41

Sliwoski, G., Kothiwale, S., Meiler, J., & Lowe, E. W. (2014). Computational Methods in Drug Discovery. *Pharmacological Reviews*, *66*(1), 334–395. https://doi.org/10.1124/PR.112.007336

Spiegel, J. O., & Durrant, J. D. (2020). AutoGrow4: An open-source genetic algorithm for de novo drug design and lead optimization. *Journal of Cheminformatics*, *12*(1), 1–16. https://doi.org/10.1186/s13321-020-00429-4

Srinivas Reddy, A., Chen, L., & Zhang, S. (2013). Structure-Based De Novo Drug Design. In *De novo Molecular Design* (pp. 97–124). Wiley-VCH Verlag GmbH & Co. KGaA. https://doi.org/10.1002/9783527677016.ch4

Stahl, M., Todorov, N. P., James, T., Mauser, H., Boehm, H.-J., & Dean, P. M. (2002). A validation study on the practical use of automated de novo design. *Journal of Computer-Aided Molecular Design 2002 16:7*, *16*(7), 459–478. https://doi.org/10.1023/A:1021242018286

Sterling, T., & Irwin, J. J. (2015). ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling*, *55*(11), 2324–2337. https://doi.org/10.1021/acs.jcim.5b00559

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. A Bradford Book.

Svetnik, V., Liaw, A., Tong, C., Christopher Culberson, J., Sheridan, R. P., & Feuston, B. P. (2003). Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. *Journal of Chemical Information and Computer Sciences*, *43*(6), 1947–1958. https://doi.org/10.1021/ci034160g

Szymkuć, S., Gajewska, E. P., Klucznik, T., Molga, K., Dittwald, P., Startek, M., Bajczyk, M., &

Grzybowski, B. A. (2016). Computer-Assisted Synthetic Planning: The End of the Beginning. In *Angewandte Chemie - International Edition* (Vol. 55, Issue 20, pp. 5904–5937). Wiley-VCH Verlag. https://doi.org/10.1002/anie.201506101

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, *1*(2), 146–160.

Thakkar, A., Chadimová, V., Bjerrum, E. J., Engkvist, O., & Reymond, J.-L. (2021). Retrosynthetic accessibility score (RAscore) – rapid machine learned synthesizability classification from AI driven retrosynthetic planning. *Chemical Science*, *12*(9), 3339–3349. https://doi.org/10.1039/D0SC05401A

The Theano Development Team, Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Snyder, J. B., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., … Zhang, Y. (2016). Theano: A Python framework for fast computation of mathematical expressions. *ArXiv E-Prints*, *abs/1605.0*.

Theilheimer, W. (1960). *Synthetic methods of Organic Chemistry. Volume 14*. Inter-science Publishers Inc.

Thomas, M., Smith, R. T., O'Boyle, N. M., de Graaf, C., & Bender, A. (2021). Comparison of structure- and ligand-based scoring functions for deep generative models: a GPCR case study. *Journal of Cheminformatics 2021 13:1*, *13*(1), 1–20. https://doi.org/10.1186/S13321-021-00516-0

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Todeschini, R., & Consonni, V. (2000). Handbook of Molecular Descriptors. In *Wiley-VCH* (Vol. 11). Wiley-VCH.

Todeschini, R., Consonni, V., Xiang, H., Holliday, J., Buscema, M., & Willett, P. (2012). Similarity coefficients for binary chemoinformatics data: Overview and extended comparison using simulated and real data sets. *Journal of Chemical Information and Modeling*, *52*(11), 2884–2901. https://doi.org/10.1021/ci300261r

Todorov, N. P., & Dean, P. M. (1997). Evaluation of a method for controlling molecular scaffold diversity in de novo ligand design. *Journal of Computer-Aided Molecular Design*, *11*(2), 175–192. https://doi.org/10.1023/A:1008042711516

Tschinke, V., & Cohen, N. C. (1993). The NEWLEAD program: a new method for the design of candidate structures from pharmacophoric hypotheses. *Journal of Medicinal Chemistry*, *36*(24), 3863–3870.

Ugi, I., Bauer, J., Bley, K., Dengler, A., Dietz, A., Fontain, E., Gruber, B., Herges, R., Knauer, M., Reitsam, K., & Stein, N. (1993). Computer-Assisted Solution of Chemical Problems—The Historical Development and the Present State of the Art of a New Discipline of Chemistry. *Angewandte Chemie International Edition in English*, *32*(2), 201–227. https://doi.org/10.1002/anie.199302011

Ugi, I., & Dugundji, J. (1973). An algebraic model of constitutional chemistry as a basis for chemical computer programs. *Top. Curr. Chem*, *39*, 19.

Ullmann, J. R. (1976). An Algorithm for Subgraph Isomorphism. *Journal of the ACM (JACM)*, *23*(1), 31–42. https://doi.org/10.1145/321921.321925

Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., Spitzer, M., & Zhao, S. (2019). Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery*, *18*(6), 463–477. https://doi.org/10.1038/s41573-019-0024-5

Van Der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2625.

Van Der Maaten, L., Postma, E., & den Herik, J. (2009). Dimensionality reduction: a comparative review. *J Mach Learn Res*, *10*, 66–71.

Varnek, A., Fourches, D., Hoonakker, F., & Solov'ev, V. P. (2005). Substructural fragments: an universal language to encode reactions, molecular and supramolecular structures. *Journal of Computer-Aided Molecular Design 2005 19:9*, *19*(9), 693–703. https://doi.org/10.1007/S10822-005-9008-0

Venkatasubramanian, V., Chan, K., & Caruthers, J. M. (1995). Evolutionary Design of Molecules with Desired Properties Using the Genetic Algorithm. *Journal of Chemical Information and Computer Sciences*, *35*(2), 188–195. https://doi.org/10.1021/ci00024a003

Verhellen, J., & Van Den Abeele, J. (2020). Illuminating elite patches of chemical space. *Chemical Science*, *11*(42), 11485–11491. https://doi.org/10.1039/D0SC03544K

Vikhar, P. A. (2017). Evolutionary algorithms: A critical review and its future prospects. *Proceedings - International Conference on Global Trends in Signal Processing, Information Computing and*

Communication, ICGTSPICC 2016, 261–265. https://doi.org/10.1109/ICGTSPICC.2016.7955308

Vinkers, H. M., De Jonge, M. R., Daeyaert, F. F. D., Heeres, J., Koymans, L. M. H., Van Lenthe, J. H., Lewi, P. J., Timmerman, H., Van Aken, K., & Janssen, P. A. J. (2003). SYNOPSIS: SYNthesize and OPti mize System in Silico. *Journal of Medicinal Chemistry*, *46*(13), 2765–2773. https://doi.org/10.1021/jm030809x

Virshup, A. M., Contreras-García, J., Wipf, P., Yang, W., & Beratan, D. N. (2013). Stochastic Voyages into Uncharted Chemical Space Produce a Representative Library of All Possible Drug-Like Compounds. *Journal of the American Chemical Society*, *135*(19), 7296–7303. https://doi.org/10.1021/ja401184g

Vléduts, G. É. (1963). Concerning one system of classification and codification of organic reactions. *Information Storage and Retrieval*, *1*(2–3), 117–146. https://doi.org/10.1016/0020-0271(63)90013-5

Vogt, M., Stumpfe, D., Geppert, H., & Bajorath, J. (2010). Scaffold hopping using two-dimensional fingerprints: True potential, black magic, or a hopeless endeavor? Guidelines for virtual screening. *Journal of Medicinal Chemistry*, *53*(15), 5707–5715. https://doi.org/10.1021/jm100492z

Voorhees, E., & Tice, D. (2000). The TREC-8 Question Answering Track Evaluation. *Proceedings of the 8th Text Retrieval Conference*.

Vowpal Wabbit. (2021). https://vowpalwabbit.org/

Wager, T. T., Hou, X., Verhoest, P. R., & Villalobos, A. (2010). Moving beyond rules: The development of a central nervous system multiparameter optimization (CNS MPO) approach to enable alignment of druglike properties. *ACS Chemical Neuroscience*, *1*(6), 435–449. https://doi.org/10.1021/cn100008c

Wallace, J. (2016). *Structure generation and de novo design using reaction networks*. University of Sheffield.

Walters, W. P. (2018). *rd_filters*. https://github.com/PatWalters/rd_filters

Wang, H., Preuss, M., Emmerich, M., & Plaat, A. (2020). Tackling Morpion Solitaire with AlphaZero-likeRanked Reward Reinforcement Learning. https://doi.org/10.48550/arXiv.2006.07970.

Wang, R., Gao, Y., & Lai, L. (2000). LigBuilder: A Multi-Purpose Program for Structure-Based Drug Design. *Molecular Modeling Annual 2000 6:7*, *6*(7), 498–516.

https://doi.org/10.1007/S0089400060498

Wang, W., Yu, G., Zhu, W., & Ju, X. (2018). Preparation method of free racemic salbutamol with high yield. In *Faming Zhuanli Shenqing*. Nanjing Zhulu Pharmaceutical Science and Technology Co., Ltd., Peop. Rep. China; Shanghai Renshang Pharmaceutical Technology Co., Ltd. .

Warr, W. A. (2011). Representation of chemical structures. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *1*(4), 557–579. https://doi.org/10.1002/WCMS.36

Warr, W. A. (2014). A short review of chemical reaction database systems, computer-aided synthesis design, reaction prediction and synthetic feasibility. *Molecular Informatics*, *33*(6–7), 469–476. https://doi.org/10.1002/MINF.201400052

Weininger, D. (1988). SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules. *Journal of Chemical Information and Computer Sciences*, *28*(1), 31–36. https://doi.org/10.1021/ci00057a005

Weininger, D. (1995). *Method and apparatus for designing molecules with desired properties by evolving successive populations* (Patent No. US5434796).

Weininger, D., Weininger, A., & Weininger, J. L. (1989). SMILES. 2. Algorithm for Generation of Unique SMILES Notation. *Journal of Chemical Information and Computer Sciences*, *29*(2), 97–101. https://doi.org/10.1021/ci00062a008

Weisgerber, D. W. (1997). Chemical abstracts service chemical registry system: History, scope, and impacts. In *Journal of the American Society for Information Science* (Vol. 48, Issue 4, pp. 349–360). https://doi.org/10.1002/(SICI)1097-4571(199704)48:4<349::AID-ASI8>3.0.CO;2-W

Wermuth, C. G., Ganellin, C. R., Lindberg, P., & Mitscher, L. A. (1998). Glossary of terms used in medicinal chemistry (IUPAC Recommendations 1998). *Pure and Applied Chemistry*, *70*(5), 1129–1143. https://doi.org/10.1351/PAC199870051129

Widmer, G. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*(1), 69–101. https://doi.org/10.1007/bf00116900

Wilcox, C. S., & Levinson, R. A. (1986). SELF-ORGANIZED KNOWLEDGE BASE FOR RECALL, DESIGN, AND DISCOVERY IN ORGANIC CHEMISTRY. *ACS Symposium Series*, 209–230. https://doi.org/10.1021/BK-1986-0306.CH018

Willett, P. (2008). From chemical documentation to chemoinformatics: 50 years of chemical information science. *Journal of Information Science*, *34*(4), 477–499.

https://doi.org/10.1177/0165551507084631

Willett, P. (2011a). Similarity searching using 2D structural fingerprints. *Methods in Molecular Biology (Clifton, N.J.)*, *672*, 133–158. https://doi.org/10.1007/978-1-60761-839-3_5

Willett, P. (2011b). Chemoinformatics: a history. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, *1*(1), 46–56. https://doi.org/10.1002/wcms.1

Willett, P., Barnard, J. M., & Downs, G. M. (1998). Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, *38*(6), 983–996. https://doi.org/10.1021/ci9800211

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3–4), 229–256. https://doi.org/10.1007/bf00992696

Winter, R., Montanari, F., Steffen, A., Briem, H., Noé, F., & Clevert, D. A. (2019). Efficient multi-objective molecular optimization in a continuous latent space. *Chemical Science*, *10*(34), 8016–8024. https://doi.org/10.1039/c9sc01928f

Wipke, W. T., & Dyott, T. M. (1974). Stereochemically Unique Naming Algorithm. *Journal of the American Chemical Society*, *96*(15), 4834–4842. https://doi.org/10.1021/ja00822a021

Wiswesser, W. J. (1952). The Wiswesser line formula notation. *Chemical and Engineering News*, *30*(34), 3523–3526. https://doi.org/10.1021/cen-v030n034.p3523

Woelfle, M., Olliaro, P., & Todd, M. H. (2011). Open science is a research accelerator. *Nature Chemistry*, *3*(10), 745–748. https://doi.org/10.1038/nchem.1149

Xu, Y., Lin, K., Wang, S., Wang, L., Cai, C., Song, C., Lai, L., & Pei, J. (2019). Deep learning for molecular generation. *Future Medicinal Chemistry*, *11*(6), 567–597. https://doi.org/10.4155/fmc-2018-0358

Yang, X., Zhang, J., Yoshizoe, K., Terayama, K., & Tsuda, K. (2017). ChemTS: an efficient python library for de novo molecular generation. *Science and Technology of Advanced Materials*, *18*(1), 972–976. https://doi.org/10.1080/14686996.2017.1401424

Yang, Z.-Y., He, J.-H., Lu, A.-P., Hou, T.-J., & Cao, D.-S. (2020). Application of Negative Design To Design a More Desirable Virtual Screening Library. *Journal of Medicinal Chemistry*. https://doi.org/10.1021/acs.jmedchem.9b01476

Yoshikawa, N., Terayama, K., Sumita, M., Homma, T., Oono, K., & Tsuda, K. (2018). Population-based De Novo Molecule Generation, Using Grammatical Evolution. *Chemistry Letters*, *47*(11), 1431–

1434. https://doi.org/10.1246/cl.180665

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. In *IEEE Computational Intelligence Magazine* (Vol. 13, Issue 3, pp. 55–75). https://doi.org/10.1109/MCI.2018.2840738

Yuan, Y., Pei, J., & Lai, L. (2011). LigBuilder 2: A Practical de Novo Drug Design Approach. *Journal of Chemical Information and Modeling*, *51*(5), 1083–1091. https://doi.org/10.1021/CI100350U

Yuan, Y., Pei, J., & Lai, L. (2020). LigBuilder V3: A Multi-Target de novo Drug Design Approach. *Frontiers in Chemistry*, *0*, 142. https://doi.org/10.3389/FCHEM.2020.00142

Zaliani, A., Boda, K., Seidel, T., Herwig, A., Schwab, C. H., Gasteiger, J., Claußen, H., Lemmen, C., Degen, J., Pärn, J., & Rarey, M. (2009). Second-generation de novo design: a view from a medicinal chemist perspective. *Journal of Computer-Aided Molecular Design*, *23*(8), 593–602. https://doi.org/10.1007/s10822-009-9291-2

Zhang, J., Terayama, K., Sumita, M., Yoshizoe, K., Ito, K., Kikuchi, J., & Tsuda, K. (2020). NMR-TS: de novo molecule identification from NMR spectra. *Science and Technology of Advanced Materials*, 552-561. https://doi.org/10.1080/14686996.2020.1793382

Zhang, T., & Yang, B. (2018). Dimension reduction for big data. *Statistics and Its Interface*, *11*(2), 295–306. https://doi.org/10.4310/SII.2018.v11.n2.a7

Zhu, J., Fan, H., Liu, H., & Shi, Y. (2001). Structure-based ligand design for flexible proteins: application of new F-DycoBlock. *Journal of Computer-Aided Molecular Design*, *15*(11), 979–996.

MarvinSketch was used for drawing and displaying chemical structures and reactions, MarvinSketch version 20.3.0, ChemAxon (https://www.chemaxon.com)

Diagrams created in Lucidchart (www.lucidchart.com)