Stochastic Modelling Of Viral Assembly Efficiency Across Capsid Geometries

Samuel Hill

PhD

University of York
Mathematics

September 2024

Abstract

Viruses are a form of obligate parasite which require the metabolism of a host cell to replicate. During infection of the host cell, viruses need to efficiently assemble new viral progeny and have evolved many strategies to achieve this. For icosahedral ssRNA viruses, capsid proteins and genomic ssRNA co-assemble without the need for subsequent packaging by a terminase. Two features that are critical for efficient assembly are the capsid's overall geometry and the arrangement of possible contacts between the RNA and capsid proteins. Thus, a natural question is whether there exist pairings of capsid geometries and arrangements of RNA contacts which result in high assembly efficiency. In this thesis, a mathematical and computational framework is developed which uses the symmetry of capsids to generate the adjacencies of all capsomers, based only on the information from one fundamental domain. This allows the enumeration of all possible icosahedral capsid geometries and RNA arrangements up to isomorphism, which can be used alongside a combinatorics program to count the possible RNA paths within a given pairing. This is implemented alongside a version of Gillespie's SSA algorithm, which stochastically models the assembly of a capsid with these geometries and RNA arrangements, generalizing the simulation of capsid assembly in ssRNA viruses. The first capsid investigated is STNV, a $T = 1$ virus. During this investigation, a set of limitations on the RNA arrangement which resulted in high efficiency assembly and repetition of arrangements was identified, showing similarity to experimental results. Next, by modelling each of the three $T = 3$ capsid geometries or tilings, significant effects on assembly for different RNA arrangements were identified, with some arrangements leading to higher assembly yields. Finally, disassembly is shown to have significant differences with assembly in the $T = 3$ class of capsid geometries, which has important consequences for understanding the uncoating process during infection.

Contents

List of Tables

List of Figures

Acknowledgements

I first arrived at the University of York in 2016, excited for my Undergraduate studies. Once here, I found wonderful communities, interesting areas of research and a lovely city that all made me want to stay. So I started a PhD and now I've spent nearly eight years as part of the University. It will be a strange feeling to finally leave.

To begin, I wish to thank my supervisors: Eric Dykeman and Reidun Twarock. You have both used your wealth of knowledge to give me lots of insight and advice over the course of my PhD, which was invaluable. I especially appreciate you reviewing this rather extended document! I also want to thank Farzad Fatehi Chenar and Rich Bingham, for many conversations and discussions.

Thanks also to my family, who have always supported me. I want to thank the friends who introduced me to (and then spent countless hours with me) playing both board games (especially *that one*) and TTRPGs: Sam Crawford, Ross Ratcliffe, Simon Hart, Berend Visser, Simen Bruinsma and Beth Dixon, amongst many more. Thanks to the rest of the PhD (and PostDoc) network, for being such a pleasant community to be a part of. I am also grateful to the many friends I have made in York's wonderful Comedy, Theatre and Climbing communities. Lastly but certainly not least, I thank Ambroise Grau for the wonderful thesis template he has produced.

Author's Declaration

I declare that the work presented in this thesis, except where otherwise stated, is based on my own research carried out at the University of York and has not been submitted previously for any degree at this or any other university. Sources are acknowledged by explicit references.
Some of the work in Chapter 6 was published as [32] and this work has also contributed to this thesis.

— 1 —

Introduction

Typically, existing computational models of virus assembly tend to either model small numbers of smaller viruses using molecular dynamics simulations [57] or model many reactions occurring simultaneously and just focus on the dynamics of specific steps in the assembly [4]. Models that do consider each of a virus' capsomers distinct and allow many capsids to exist simultaneously tend to only cover small viruses [19]. Over this thesis, models that can model the assembly of many copies of larger viruses, whilst still considering each capsomer, are developed. Some techniques are new and others develop on expanding on existing work. Notably, the results presented in Chapter 6 which were observed by using techniques described in Chapters 4 and 5 were published [32].

Viruses are infectious biological agents. They cannot replicate on their own, so they infect living organisms and use their host cell's machinery to replicate. For each of the biological kingdoms of life, there are viruses that will infect the members of that kingdom, from bacteria to animals to plants to fungi.

When a virus infects an organism, this often causes disease, with viruses typically being parasitic. However there are some viruses that form symbiotic relationships with their hosts, though these are less common. Parasitic viruses can cause issues for humans when viruses infect our crops, our livestock and even ourselves.

Historical examples of viruses include Measles and Smallpox, which are thought to have been infecting people since 400BC and 1500 BC respectively [51, 15]. Between 1855 and 2005, measles is believed to have killed ≈ 200 millions people [40]. In the 20^{th} century alone, Smallpox is thought to have killed up to 300 million people prior to its eradication in 1980 [31]. Some viruses are better known for large outbreaks in specific years, such as the 1918–1920 Flu Pandemic caused by Influenza shortly

after the events of WW1, which is estimated to have killed approximately 50 million people [11]. Another example is the HIV/AIDs pandemic, which began in 1981 and has killed over 40 million people [55]. Lastly, COVID-19 is another virus that was discovered in 2019 and resulted in a pandemic that began in 2020 and has caused over 7 million confirmed deaths, as of the 6th Jun 2024 [38].

In addition to the massive scale of human suffering caused by viruses, they infect plants and other animals. Modern examples of viral outbreaks with huge impacts on agriculture include the Papaya Ringspot Virus, which had an outbreak in the 1990s and nearly destroyed the papaya industry in Hawai'i [53]. Also, an outbreak of Maize Lethal Necrosis Disease (resulting from a co-infection of a few different viruses) caused a 10% decrease in the yield of maize grown in Kenya in the 2014-15 growing season [56]. It is not just plants that can get infected, there are also viruses that infect livestock, for example Foot and Mouth Disease. In the 2007 Foot and Mouth outbreak in the UK, over 2000 animals were slaughtered to prevent further infections and there have been many more outbreaks of this disease across many different continents [10]. On top of outbreaks, viruses have other impacts on humanity, with some simply reducing crop growth, some making people unwell for periods of time and many other effects.

Thus, learning about the viral replication cycle is important, to help reduce the effects of future outbreaks and diseases. The aspect of this cycle that will be covered during this thesis is assembly, after a virus has infected a cell. During assembly, the infected cell is forced to produce copies of the virus' genome and many copies of the virus' Coat Proteins (CPs). Then, these proteins will form the capsid, a protein shell that contains the genome, before these completed virions are released from the host cell. This step features the genome, released from its protective capsid shell, so is a time where drugs could be administered, to reduce the efficacy of the virus.

Initially, Chapter 2 will cover some methods that can be used to model chemical kinetics. There are a number of ways to do this, including using ODE-based methods, where large concentrations are treated as continuous. It also covers Stochastic Methods, primarily Gillespie's SSA algorithm (or improvements to it), where the number of proteins and capsids present are treated as being discrete. Lastly, this chapter covers some relevant virus biology, to assist in modelling their assembly.

Next, Chapter 3 will cover methods of modelling virus assembly that use Equilibrium and Quasi-Equilibrium Models. One involves calculating effective reaction rates for two-step reactions via Markov Chains and using this to calculate the equilibrium

point of a simple viral system. The other takes a system made up of a small number of second order differential equations and then uses partition theory to generate a list of states and adapt the system, so it is made up of many first order differential equations. Once the differential equations were all first order, it was simple to identify the quasi-stable states in the system.

Next, Chapter 4 describes a computational model. This provides a method that allows simple and convenient input of the capsid's information, which is then generalised to the entire capsid using the icosahedral symmetry inherent to the capsid. Whether these are large or small capsids, the input has a similar complexity. Once generated, the capsid's information is stored as a pair of graphs. In addition to this, the chapter specifies how our stochastic simulation works, alongside some information about the dynamics of the viral assembly process.

Once a method to model the assembly of viral capsids has been described, a means to analyse the results of these simulations is needed. Chapter 5 initially covers a combinatoric approach to how the RNA can sit inside the capsid. Then it takes the output of the stochastic simulations and performs analysis on how the RNA sits on the interior of the capsid at the end of the simulation. Following from this, methods to inspect the intermediate structures that the capsid form during assembly were also developed. Lastly, visualisation techniques are described.

Next, the methods described in the previous two chapters were used. In Chapter 6, a small virus called Satellite Tobacco Necrosis Virus (STNV) is modelled. As there is a wealth of experimental data available for this virus, these are used to augment the modelling and test to see if the model is accurate.

After working on STNV, Chapter 7 considers a class of larger viruses. These viruses have three different tilings, with differently shaped capsomers, so instead of focusing on any specific viruses, this chapter investigates the effects of changing geometries on the assembly.

Most of this thesis has focused upon the assembly of viruses. However, it is not the only step of a virus' replication cycle that can be modelled using biochemical reactions. In Chapter 8, the process of disassembly is briefly covered. Then, using a small modification to the stochastic simulation, virus disassembly methods are developed.

— 2 —

Modelling Reaction Kinetics In Biological Systems

In this chapter, the background of the mathematical and biological concepts that are used throughout this thesis will be presented. It will highlight the uses and restrictions of ODE based models, cover the advantages and disadvantages of using Stochastic based models and give details of how viral assembly occurs.

2.1 ODE BASED MODELS

Chemical Kinetics is an important field, for both chemical and biological systems. It allows for modelling of many systems, from factories to test tubes to cells. However different systems require different modelling techniques at different levels of coarse-graining (and adapted to distinct expected settings), each giving a different precision.

To get the highest accuracy in these simulations, molecular dynamics simulations can be used. However, they take a long time to run, even for small systems over small time periods. All-atom simulations of viruses have been performed for very short time periods, *e.g.* the $T = 4$ virus HBV, which contains 240 proteins and consists of 5.9 million atoms [44]. Simpler MD models exist, where each protein can be treated as a separate particle, which allows much faster computation with the limitation that assumptions must be made regarding how proteins can interact [57].

The main method used for large systems is known as the Rate Reaction Equation (RRE), where there are sufficiently many molecules that it is not necessary to simulate each one individually. It requires setting the system up as a series of ODEs and can

be written as:

$$\frac{dX_i}{dt} = f_i(X_1, X_2, \ldots, X_N) \quad i = 1, \ldots, N, \tag{2.1}$$

which represents a system with N different species. Here, X_i is the number of molecules of species i in the system and f_i can be determined using knowledge of the reactions involved in the system. The equations show the rate of change in the number of molecules of each species present. For systems that only include first order reactions, these equations are trivially solvable by creating a matrix of rates A such that $\dot{\mathbf{X}} = A\mathbf{X}$. Its solution is given by:

$$\mathbf{X}(t) = \sum_{i=1}^{N} C_i e^{\lambda_i t} \hat{\mathbf{x}}_i, \tag{2.2}$$

where λ_i and $\hat{\mathbf{x}}_i$ are the eigenvalues and eigenvectors of A, respectively. C_i is a constant to be determined from the initial conditions. For second order systems, analytical solutions can sometimes be found. However, this is usually only possible in the simplest of systems. For more complex ones, ODE solvers such as the classical Runge-Kutta method are employed. The RRE can also be written to indicate the probability of a given species' presence, as an equivalent formulation. The above equations assume that the system is at a constant temperature and that it is well mixed. This method is deterministic and continuous, so it neglects individual reactions and gives an idea of how the system will advance over time. This is accurate in systems as small as a test tube, as there are sufficiently many particles present in the system that treating them as continuous is appropriate. The main issue is that at the start of assembly, the number of reactants is small, so this method will not be accurate.

Additionally, virus assembly takes many discrete steps from free capsomers to completed capsids and can follow a large number of different pathways to assembly, especially compared to the number of particles present in a cell, so there will typically be few particles in each state, which means letting it be treated as a continuous system is not appropriate. Viruses also tend to rely on second order reactions during assembly, so the trivial solution to the RRE model would not be available, even if the RRE was appropriate to use. Additionally, another issue ODE models have is that they need each state and each reaction to be predefined, which means that there is either a very long process to count and compile a huge number of rate equations, or the model has to neglect a large number of the intermediates that could be formed during virus assembly. An example of a model, which neglects all but the lowest

energy intermediates for a dodecahedral capsid, that was proposed by Zlotnick [59], will be covered in Chapter 3. To model larger and more complex viruses' assembly, a model where particles are treated as discrete is needed. This leads to a need for a stochastic formulation, that does not need to rely on all states being predefined. This is covered in the next section.

2.2 STOCHASTIC BASED MODELLING

This section will discuss many stochastic methods of modelling chemical kinetics, based off Gillespie's SSA method, first published in 1976 [27]. It will follow somewhat the review paper by Gillespie [29].

2.2.1 STOCHASTIC SIMULATION ALGORITHM

Here, the Gillespie Stochastic Simulation Algorithm (SSA) is introduced [27]. This assumes that the system has a constant volume and is at thermal equilibrium. It also relies on the assumption that most collisions are elastic, which leads to the positions and velocities of the particles being randomised (which means the system is well mixed). We begin by assuming that the system contains N chemical species (S_i), which would be chemicals in a chemical reaction or intermediate structures in virus assembly. These species have M different reactions (R_i) that can occur between them. If any of these reactions are reversible, then each direction of reaction will be treated as a separate reaction, therefore the notation $\rightarrow$ will be used instead of $\rightleftharpoons$. The state is determined by

$$\mathbf{x}(t) = (x_1(t), x_2(t), \ldots, x_N(t)), \tag{2.3}$$

where x_i is the number of particles of type S_i present. A reaction R_i can be described by two terms, $\mathbf{v}_i$ and a_i. The first is a vector, such that if reaction R_i occurs, then the system will change from $\mathbf{x}$ to $\mathbf{x} + \mathbf{v}_i$, where every element of $\mathbf{v}_i$ will be a positive or negative integer, depending on how many reactants are used and how many products are produced in the reaction. The second, $a_i(\mathbf{x})$ is a propensity function, which can be defined that at time t, $a_i(\mathbf{x})dt$ is the probability that R_i will occur before $t + dt$. The propensity function is also able to depend explicitly on t. However, we assume here that it depends only on the state of the system (and so only depends on t implicitly).

The values of $a_i(\mathbf{x})$ can be calculated with knowledge of the reactions. If R_i is a first order reaction (*e.g.* $S_j \rightarrow \ldots$), then $a_i(\mathbf{x}) = c_i x_j$, as it just depends on how fast the reaction occurs (the rate constant c_i) and the number of particles that can undergo that reaction. For a second order reaction of two different species (*e.g.* $S_j + S_k \rightarrow \ldots$), then $a_i(\mathbf{x}) = c_i x_j x_k$, as there are $x_j x_k$ different combinations of pairs of particles to react. This second order rate constant (c_i) is slightly different to the previous one, as it is the rate at which these pairs will collide and react together. When there is a second order reaction between two particles of the same species (*e.g.* $S_j + S_j \rightarrow \ldots$), then $a_i(\mathbf{x}) = \frac{c_i}{2} x_j(x_j - 1)$ with the same rate constant as the other second order reaction and the $\frac{x_j(x_j-1)}{2}$ combinations of two reactant particles. In biological systems, reactions tend to be only first or second order. In general chemistry, third (or higher) order reactions (*i.e.* reactions that have three or more reactants) do exist but tend to have a short-lived intermediate from two reactants colliding before the third joins and so can be thought of as the result of two (or more) successive second order reactions.

For a model system of three species, $(S_1, S_2, S_3) = (A, B, C)$ and three reactions, $R_1 : A + B \rightarrow C$, $R_2 : A + A \rightarrow B$ and $R_3 : C \rightarrow 3A$, we can find $\mathbf{v}_i$ and a_i for each i. Reaction R_1 is second order, with two different reactants, so $\mathbf{v}_1 = (-1, -1, +1)$ and $a_1 = c_1 x_1 x_2$. The second reaction, R_2, is also second order, but the reactants are from the same species, so $\mathbf{v}_2 = (-2, +1, 0)$ and $a_2 = \frac{c_2(x_1-1)}{2} x_1$. The last reaction, R_3, is first order, so has $\mathbf{v}_3 = (+3, 0, -1)$ and $a_3 = c_3 x_3$.

Every c_i relates to the reaction rate constant k_i from deterministic chemical kinetics. For first order reactions, $c_i = k_i$. For second order reactions that involve two particles of different species, $c_i = \frac{k_i}{V}$. For second order reactions that involve two particles of the same species, $c_i = \frac{2k_i}{V}$. In these equations, V is the volume of the system.

To simulate a system, the two key pieces of information that are needed are which reaction is going to happen next and when it will happen. With these two pieces of information, the entirety of a system's reactions can be modelled. To do this, we define $p(\tau, i|\mathbf{x}, t)$ to be the probability that the next reaction is an R_i reaction that will occur between time $t + \tau$ and $t + \tau + d\tau$, given the system is in state $\mathbf{x}$ at time t. This is sometimes called a reaction probability density function for the space of $0 \leq \tau < \infty$ and $i = 1, 2, \ldots, M$. We can state this probability as the product of the probability no reaction will happen in the interval $(t, t + \tau)$, defined as $p_0(\tau|\mathbf{x}, t)$ with the probability that reaction R_i will occur in the interval $(t + \tau, t + \tau + d\tau)$,

which is $a_i d\tau$. In a time period where no reaction occurs, $a_i(\mathbf{x}) = a_i$ is a constant. Together this gives:

$$p(\tau, i|\mathbf{x}, t) = p_0(\tau|\mathbf{x}, t) \cdot a_i d\tau. \tag{2.4}$$

Following from this, consider $p_0(\tau' + d\tau'|\mathbf{x}, t)$. This is equal to the product of the probability that no reaction will happen in the interval $(t, t + \tau')$ and the probability that no reaction will happen in the interval $(t + \tau', t + \tau' + d\tau')$, so

$$p_0(\tau' + d\tau'|\mathbf{x}, t) = p_0(\tau'|\mathbf{x}, t) \cdot (1 - \sum_{\nu=1}^{M} a_\nu d\tau'). \tag{2.5}$$

Rearranging this gives

$$\frac{p_0(\tau' + d\tau'|\mathbf{x}, t) - p_0(\tau'|\mathbf{x}, t)}{d\tau'} = -p_0(\tau'|\mathbf{x}, t) \sum_{\nu=1}^{M} a_\nu = \frac{dp_0(\tau'|\mathbf{x}, t)}{d\tau'}. \tag{2.6}$$

Defining $a_0 := \sum_{\nu=1}^{M} a_\nu$, this can be solved to give $p_0(\tau|\mathbf{x}, t) = e^{-a_0\tau}$ and therefore

$$p(\tau, i|\mathbf{x}, t) = a_i \cdot e^{-a_0\tau}. \tag{2.7}$$

For this to be useful in a simulation, we need to be able to sample $p(\tau, i|\mathbf{x}, t)$, to randomly select which reactions will occur and when. This can be done with the Inversion Generating Method.

2.2.2 Inversion Generating Method For Sampling

Let $P : [a, b] \to \mathbb{R}$ be a probability density function, $P(x)$, where $P(x')dx'$ is the probability that x is between x' and $x' + dx'$. Thus, we can define

$$F(x_0) = \int_{-\infty}^{x_0} P(x')dx' \tag{2.8}$$

as the probability that $x \leq x_0$. This results in $F(-\infty) = 0$ and we require $F(\infty) = 1$ for $P(x)$ to be a probability density function. As a result, we have $0 \leq F(x) \leq 1$, which is also the range of a uniform random number in the unit interval. So, if we choose x such that $F(x) = r$, we can use

$$x = F^{-1}(r) \tag{2.9}$$

to generate this random value of x.

To prove that this is true, we must calculate the probability that a random value of x is between x' and $x' + dx'$. This is equal to the probability that r lies between

$F(x')$ and $F(x' + dx')$, which is simply $F(x' + dx') - F(x')$. Using the definition of a differential, this can be written as $F(x' + dx') - F(x') = F'(x')dx'$. Recalling Equation 2.8, we can write

$$F(x' + dx') - F(x') = F'(x')dx' = P(x')dx', \tag{2.10}$$

which is the desired probability, so Equation 2.9 gives us the desired values for x.

A similar argument can be made for discrete functions. We desire integer i, from probability function $P(i)$, where $P(i')$ is the probability that $i = i'$. We choose a uniform random number $0 \leq r \leq 1$. Now, let

$$F(i) = \sum_{i'=-\infty}^{i} P(i') \tag{2.11}$$

be the probability that $i \leq i'$. As before, we have $F(-\infty) = 0$ from the definition and $F(\infty) = 1$ if $P(i)$ is a valid probability function.

We should take i, such that it satisfies

$$F(i-1) < r \leq F(i). \tag{2.12}$$

This is also satisfied by choosing the smallest i that satisfies $F(i-1) < r$. Again, we need proof that this results in the desired distribution of i. So we inspect the probability that $i = i'$, which is the probability that r is between $F(i')$ and $F(i'-1)$. As r is a uniform random number, this is just $F(i') - F(i'-1)$, which rearranges to give $\sum_{i''=-\infty}^{i'} P(i'') - \sum_{i''=-\infty}^{i'-1} P(i'') = P(i')$. Which was defined as the probability that $i = i'$, so Equation 2.12 gives us a method to calculate i.

Now we need to apply these to the probability distribution given in Equation 2.7. To begin, we want the τ for the next reaction that is likely to happen, not just τ for a given reaction, so we define

$$P(\tau) = \sum_{i=1}^{M} p(\tau, i|\mathbf{x}, t) = a_0 \cdot e^{-a_0\tau}. \tag{2.13}$$

Next, this gives us

$$\begin{aligned} F(\tau_0) &= \int_0^{x_0} a_0 e^{-a_0\tau'} d\tau' \\ &= -[e^{-a_0\tau'}]_0^{\tau_0} \\ &= 1 - e^{-a_0\tau}. \end{aligned}$$

Setting this as equal to a random number r, we get $r = 1 - e^{-a_0\tau}$, which can be rearranged to give $1 - r = e^{-a_0\tau}$. Taking logarithms of both sides gives $\ln(1-r) = -a_0\tau$, so we can solve for τ to give:

$$\tau = \frac{1}{a_0} \ln\left(\frac{1}{r}\right). \tag{2.14}$$

(substituting $1 - r$ for r, as they give equivalent distributions in the unit interval).

To cover which reaction occurs, we consider all time, so

$$\begin{aligned} P(i) &= \int_0^\infty p(\tau, i|\mathbf{x}, t)d\tau \\ &= a_i \int_0^\infty e^{-a_0\tau} d\tau \\ &= \frac{a_i}{a_0}. \end{aligned}$$

Thus, $F(i) = \frac{1}{a_0}\sum_{i'=1}^{i} a_{i'}$. So r will be in the interval:

$$\sum_{i'=1}^{i-1} a_{i'} < ra_0 \leq \sum_{i'=1}^{i} a_{i'}. \tag{2.15}$$

2.2.3 ALGORITHM FOR THE DIRECT METHOD

As shown above, the following equations will generate τ and i with the correct distributions, as given in Equation 2.7.

$$\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right) \tag{2.16}$$

$$\sum_{i'=1}^{i} a_{i'}(\mathbf{x}) > r_2 a_0, \tag{2.17}$$

where r_1 and r_2 are uniform random numbers in the unit interval and i is the smallest integer that satisfies the above condition.

These give us the SSA Algorithm (Direct Method):

0. Initialise $t = t_0$ and $\mathbf{x} = \mathbf{x_0}$
1. At state $t, \mathbf{x}$, evaluate all $a_i(\mathbf{x})$ and a_0
2. Generate τ, i
3. Advance $t \leftarrow t + \tau$ and $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}_i$
4. Repeat from Step 1 until $t \geq t_{max}$

This can be slow, due to the algorithm calculating every reaction. However, this does ensure that the modelling is accurate. This is called the SSA direct method.

2.2.4 First Reaction Method

One alternative to the Direct Method is called the First Reaction Method [28]. For this method, M uniform random numbers in the unit interval are generated, $r_1, r_2, \ldots, r_M$. Then, these are used to calculate:

$$\tau_{i'} = \frac{1}{a_{i'}} \ln\left(\frac{1}{r_{i'}}\right) \tag{2.18}$$

for all $i' = 1, 2, \ldots, M$. This gives $\tau_{i'}$, the time until the next $R_{i'}$ reaction, for all reactions. Then, $\tau = \min(\tau_{i'})$ and $i = i''$, where $\tau_{i''} = \min(\tau_{i'})$. This procedure is less efficient than the Direct Method in systems where there are a large number of reactions but gives a distribution that is equivalent to the Direct Method.

Assuming there are no other reactions in the interval $(t, t+\tau)$, then the probability of reaction R_i in the interval $(t+\tau, t+\tau+d\tau)$ is $P_i(\tau)d\tau = e^{-a_i\tau} \cdot a_i d\tau$ as no other reactions will have affected the quantity of reactants present in the system.

Let $\tilde{P}(\tau, i)d\tau$ be the probability that the procedure above will result in an R_i reaction in the interval $(t+\tau, t+\tau+d\tau)$, so

$$\tilde{P}(\tau, i)d\tau = \text{Prob}\{\tau < \tau_i < \tau + d\tau\} \cdot \text{Prob}\{\tau_\nu > \tau, \forall \nu \neq i\}, \tag{2.19}$$

which is the product of the probability that reaction R_i occurs in the interval $(t+\tau, t+\tau+d\tau)$ and the probability that reaction R_i is the first reaction to occur. Here,

$$\text{Prob}\{\tau < \tau_i < \tau + d\tau\} = e^{-a_i\tau} \cdot a_i d\tau. \tag{2.20}$$

Next, considering the second probability,

$$\begin{aligned}
\text{Prob}\{\tau_\nu > \tau, \forall \nu \neq i\} &= \text{Prob}\left\{\frac{1}{a_\nu} \ln\left(\frac{1}{r_\nu}\right) > \tau, \forall \nu \neq i\right\} \\
&= \text{Prob}\left\{r_\nu < e^{-a_\nu \tau}, \forall \nu \neq i\right\} \\
&= \prod_{\nu=1, \nu\neq i}^{M} \text{Prob}\{r_\nu < e^{-a_\nu\tau}\}
\end{aligned}$$

and recalling that r_ν is a uniform random number in the unit interval, we can write

$$\text{Prob}\{\tau_\nu > \tau, \forall \nu \neq i\} = \prod_{\nu=1, \nu\neq i}^{M} e^{-a_\nu\tau}. \tag{2.21}$$

Thus,

$$\begin{aligned}\tilde{P}(\tau, i)d\tau &= \prod_{\nu=1}^{M} e^{-a_\nu \tau} \cdot a_i d\tau \\ &= a_i d\tau e^{\sum_{\nu=1}^{M} -a_\nu \tau} \\ &= a_i d\tau \cdot e^{-a_0 \tau} \\ &= P(\tau, i)d\tau,\end{aligned}$$

which is the original distribution and therefore the methods are equivalent.

2.2.5 Improvements To The Direct Method

The first improvement to the Gillespie algorithm is the Next Reaction method [26]. It is similar to the First Reaction method, as it starts by generating a $\tau_i = \frac{1}{a_i} \ln\left(\frac{1}{r_i}\right)$ for each reaction. However, unlike in the First Reaction method, it only requires one new random number for each step and it uses absolute time. Hence, if $\tau_j = \min(\tau_i)$, then the next reaction will be an R_j reaction at time $t = \tau_j$. It stores each of the τ_i and a_i values and only recalculates a_i when a reaction that affects a_i takes place. It does this by creating, for each reaction R_i, a list of which a_is are affected by said reaction, during the initialisation.

When a_i has been recalculated, τ_i is adjusted to account for this change. Previously, we used the function, $F_i(\tau) = \int_0^\tau a_i e^{-a_i \tau'} d\tau'$ to transform a random number into a time for the reaction to occur. It relied on the fact that no reactions would occur in the interval $(t, t+\tau)$ and so that all a_i would be treated as constants. In the Next Reaction method, this assumption is not used, so we have from Equation 2.8 the probability that, at time t, an R_i reaction would occur by time τ:

$$F_i(\tau) = \int_t^\tau a_i(\tau') e^{-a_i(\tau')\tau'} d\tau'. \tag{2.22}$$

Here, $a_i(t)$ is a piecewise function, so the integral can be simplified into a sum of integrals where a_i can be treated as constant, as before. This allows the same random number to be used for the reaction until the reaction next occurs. The algorithm does this by changing every τ_i where the a_i it depended upon has changed, so that $\tau_i^{\text{new}} = \frac{a_i^{\text{old}}}{a_i^{\text{new}}}(\tau_i^{\text{old}} - t) + t$, which is the new time that the next R_i reaction is due to occur. These τ_i values are stored in a tree, such that each parent is smaller than either of its children, allowing faster searching for the minimum τ_i value and allowing

efficient updates. This is more challenging to code than the direct method but does allow a greater efficiency to the method.

Another improvement to the Direct Method is the Modified Direct Method [8]. This method operates along the same lines as the Direct Method but it includes an extra step during the initialisation process. It initialises and does a prerun to compare the relative sizes of all of the values of a_i. Then, it re-indexes the reactions so that reactions with larger a_i values during the prerun are given smaller indices, as this reduces the number of comparisons done for Equation 2.17. This allows the method to become competitive with the Next Reaction Method. It works best when there are many reaction channels, which have a wide range of a_i values, such as those often found in biological systems.

The Modified Direct Method has been improved upon with the Sorting Direct Method [39], where the indices of the reactions are modified dynamically as the simulation evolves. It does this without a prerun and continues to do this as the simulation progresses, by exchanging the index of R_i with the index of R_{i-1}, whenever an R_i reaction occurs. This allows it to account for values of a_i that change over the course of the simulation and means that the reactions that fire most often will rise to the top of the list, which will result in fewer comparisons for Equation 2.17.

There are numerous other methods that will improve upon the efficiency of the Direct Method for SSA but they all follow the same mathematical background as detailed in Section 2.2.1.

2.2.6 Tau Leaping

The exact SSA methods are very good for giving precise trajectories for chemical systems, as they only allow single reactions and so are precise. However, these exact methods have issues with efficiency for larger systems with large quantities of reactants. Some improvements to efficiency have been given above. However, the fact that these algorithms all model every reaction means there is a limit to the maximal efficiency that any algorithm can give. And often, these trajectories contain an unnecessary excess of detail. Thus, Gillespie suggested an alternate, approximate method called τ-leaping [30].

This method allows the system to model many reactions occurring simultaneously in a larger time interval. It works best for systems that have a relatively small number of species and reaction pathways compared to the number of reactants

present in the system, *i.e.* each species is well populated.

We define $Q(k_1, \ldots, k_M|\tau; \mathbf{x}, t)$ as the probability that a system in state $\mathbf{x}$ at time t will have exactly k_j firings of reaction R_j for all $j = 1, \ldots, M$ before time $t + \tau$. We can also write $K_j(\tau; \mathbf{x}, t)$ as the number of times the system in state $\mathbf{x}$ at time t will fire reaction R_j before time $t + \tau$. To calculate $Q(k_1, \ldots, k_M|\tau; \mathbf{x}, t)$ is challenging, so a condition is required to simplify the calculation. This condition, called the Leap Condition, requires the value of τ to be small enough that the changes in reactant quantities are also small, which means that the values of a_j will not change appreciably during a leap. This means that $a_j dt$ will give the probability that reaction R_j will occur during any interval dt on the interval $(t, t + \tau)$. Thus, the number of firings for a constant probability is a Poisson process and

$$K_j(\tau; \mathbf{x}, t) = \mathcal{P}(a_j \tau) \tag{2.23}$$

is a Poisson random variable, for $j = 1, \ldots, M$. Note that each $K_j(\tau; \mathbf{x}, t)$ must be independent of every other $K_j(\tau; \mathbf{x}, t)$, so it follows that:

$$Q(k_1, \ldots, k_M|\tau; \mathbf{x}, t) = \prod_{j=1}^{M} P_{\mathcal{P}}(k_j; a_j, \tau). \tag{2.24}$$

This means that, in simplest terms, we can write each leap as $\mathbf{x}(t + \tau) = \mathbf{x}(t) + \sum_{j=1}^{M} k_j \mathbf{v}_j$, where k_j is a sample of K_j. Choosing τ is important, as with a smaller value, the simulation is more precise but loses efficiency. Larger values of τ give less precision at a higher speed. When each reactant species is well populated, the Leap Condition can be satisfied even with a large number of reactions occurring in each leap. The smallest value of τ that should be taken is $\tau = \frac{1}{a_0(x)}$, as a smaller value would lead to many leaps not including a single reaction and even at this value, many leaps would only contain one reaction. This would lead to these leaps being exact. However, it would take longer to run simulations than running the direct method. Earlier methods for τ-leaping had issues whereby they would leap too far and result in negative quantities of reactants, which is not physical, so additional checks were added to avoid this. These checks slowed down the simulations, so methods which performed these checks efficiently were created.

An algorithm for how to run a τ-leaping simulations with an efficient step size selection [7] is as follows:

1 Create a list of all reactant species in the simulation, I_{RS}. Then, find all critical reactions, which are defined as any reaction R_j, where n_c firings of

the reaction would exhaust a reactant (n_c is usually set somewhere between 2 and 20) and where the reaction also has a non-zero propensity function a_j. Store the indices for all critical reactions in J_{CR} and all of the indices for all non-critical reactions in J_{NCR}.

2 Using an error control bound $0 < \epsilon \ll 1$, we need to calculate τ', the largest allowed leap for non-critical values.

2.1 For each $i \in I_{RS}$, inspect all reactions to find the highest order of a reaction where species S_i is a reactant and store it in an array $HOR(i)$.

2.2 Find g_i where:

2.2.1 If $HOR(i) = 1$, then $g_i = 1$

2.2.2 If $HOR(i) = 2$, then $g_i = 2$

2.2.3 If $HOR(i) = 2$ and the reaction is between two S_i molecules, then $g_i = \left(2 + \frac{1}{x_i - 1}\right)$

2.3 Calculate $\hat{\mu}_i(\mathbf{x}) = \sum_{j \in J_{NCR}} v_{ij} a_j$ for all $i \in I_{RS}$

2.4 Calculate $\hat{\sigma}_i^2(\mathbf{x}) = \sum_{j \in J_{NCR}} v_{ij}^2 a_j$ for all $i \in I_{RS}$

2.5 Calculate $\tau' = \min\limits_{i \in J_{NCR}} \left\{ \frac{\max\{\epsilon x_i / g_i, 1\}}{|\hat{\mu}_i(\mathbf{x})|}, \frac{\max\{\epsilon x_i / g_i, 1\}^2}{\hat{\sigma_i}^2(\mathbf{x})} \right\}$

3 If $\tau' \leq \frac{n_m}{a_0}$ for some small integer $n_m \approx 10$, then perform some (≈ 100) steps of the SSA method and then go to Step 1.

4 Compute $a_0^c = \sum_{j \in J_{CR}} a_j$ and set $\tau'' = \frac{1}{a_0^c} \ln\left(\frac{1}{r}\right)$, for some uniform random number in the unit interval r, as an estimate of the time until the next critical reaction.

5 Let $\tau = \min\{\tau', \tau''\}$

5.1 If $\tau' < \tau''$, then no critical reactions will occur in this leap, so set $k_j = 0$ for all $j \in J_{CR}$. For non-critical reactions, set $k_j = \mathcal{P}(a_j \tau)$ for all $j \in J_{NCR}$.

5.2 If $\tau'' < \tau'$, then calculate j_c, such that it is the smallest element of J_{CR} that satisfies $\sum\limits_{j' \in J_{CR}, j' < j_c} a_{i'} \leq r a_0^c$, for some uniform random number in the unit interval r. Let $k_{j_c} = 1$ and $k_j = 0$ for all other $j \in J_{CR}$. For non-critical reactions, set $k_j = \mathcal{P}(a_j \tau)$ for all $j \in J_{NCR}$.

6 If any element of $\mathbf{x} + \sum_{j=1}^{M} k_j \mathbf{v}_j$ is less than zero, then set $\tau' = \frac{\tau'}{2}$ and go to Step 3. Otherwise, set $t = t + \tau$ and $\mathbf{x} = \mathbf{x} + \sum_{j=1}^{M} k_j \mathbf{v}_j$. If the end time for the simulation has not been reached, go to Step 1, else stop.

This allows for fast simulation of chemical kinetics in systems when all species are well populated. Unfortunately, this is usually not the case during virus assembly, as there is a huge number of potential species compared to the number of intermediates in the average cell, so τ-leaping will not be used in these simulations.

2.3 VIRUS ASSEMBLY BIOLOGY

Viruses are infectious agents that are not able to replicate on their own. They must infect a host and use the host cell's machinery to replicate. Whilst many viruses are parasitic, some are able to form a symbiotic relationship with their host.

Viruses are typically made up of at least two features: their genetic information and a shell of proteins that contains the genetic information, called a capsid. Some viruses have other features but all have these two. The genetic information is stored as either DNA or RNA, both of which can be either single-stranded (ss) or double-stranded (ds) [37, 3]. The most common way to store the genetic information is as ssRNA, which are the viruses focused on in this thesis [3]. There are multiple different shapes of capsids, including rod shaped helical capsids, icosohedral capsids and complex viruses. Most viruses have icosahedral capsids, also known as spherical and these will be the main focus in this thesis [37, 3].

Icosahedral viruses take the approximate shape of an icosahedron, a 20-sided shape made up of equilateral triangles. Their capsids are made up of many copies of the same capsid (or coat) protein (CP). For small viruses, all of the CPs in the capsid sit in one of 60 rotationally equivalent positions. They do this due to the principle of genetic economy, which determines that it is more efficient to code a small protein once and use many copies of it, than to code for one large protein and only use it once [33]. This suggests all proteins must sit in rotationally equivalent positions, which leads to the need for the capsid to emulate one of the five platonic solids: tetrahedron, cube, octahedron, dodecahedron and icosahedron. However, the cube and octahedron share a symmetry group, as do the dodecahedron and icosahedron, so there are only three choices of symmetry group. Viruses tend to use the icosahedral (or dodecahedral) symmetry as it has the largest number of rotationally equivalent positions and so allows the largest capsid to be built from the same sized protein. An example of an icosahedral virus formed from 60 proteins, STNV, is shown in Figure 2.1a.

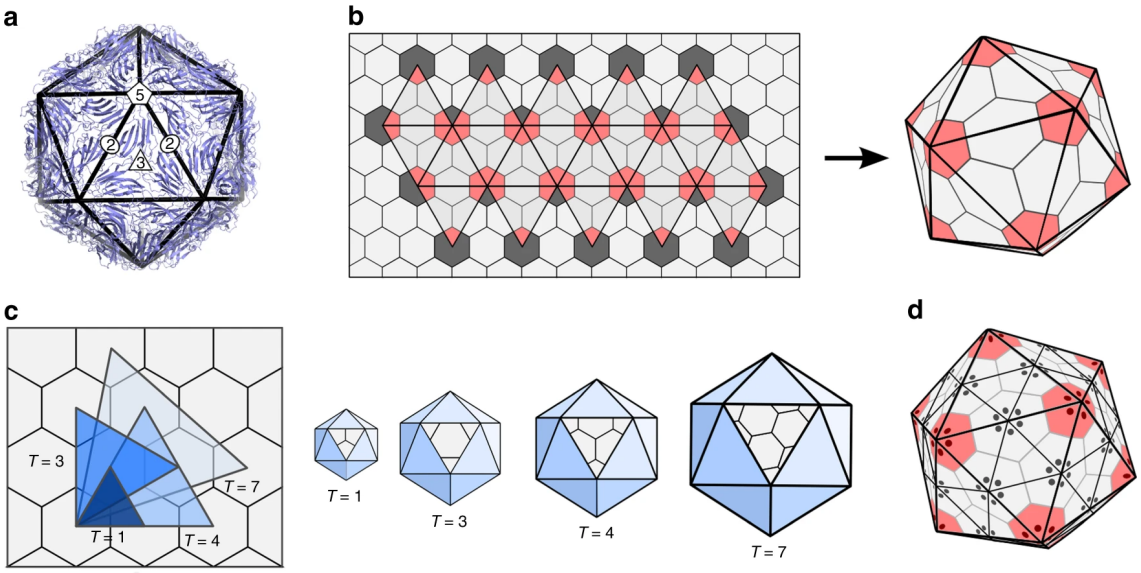

Figure 2.1: Figure and Caption adapted from [54]. Capsid architecture according to Caspar and Klug theory. (a) Viruses exhibit the characteristic 5-, 3- and 2-fold rotational symmetry axes of icosahedral symmetry, indicated here with reference to the vertices, edges and faces of an icosahedral frame superimposed on the crystal structure of the $T = 1$ STNV shell (figure based on PDB-ID 2BUK). (b) Construction of an icosahedral polyhedron via replacement of hexagons in a hexagonal lattice by the equivalent of 12 equidistant pentagons (red). Dark grey areas indicate parts of hexagons in the lattice that do not form part of the surface lattice of the final polyhedral shape. (c) One of the 20 triangles of the icosahedral frame is shown superimposed on the hexagonal grid for the four smallest polyhedra that can be constructed in CK theory. These are shown in increasingly lighter shades of blue: $T(1,0) = 1$, $T(1,1) = 3$, $T(2,0) = 4$ and $T(2,1) = 7$. The corresponding polyhedra (right) have 20 identical triangular faces corresponding to the triangles (left), one of which is shown in each case. (d) The CP positions are indicated with reference to the dual polyhedron, that is, the triangulated structure obtained by connecting midpoints of adjacent hexagons and pentagons in the surface lattice. CPs are positioned in the corners of the triangular faces (shown here as dots) and result in clusters (capsomers) of six (hexamers) and five (pentamers) CPs. The example shown corresponds to a $T = 4$ layout, formed from 240 CPs that are organised as 12 pentamers (red) and 30 hexamers.

However, to generate larger viruses, more copies of the CPs needs to be used. Work by Caspar and Klug [9] determined that the CPs do not need to be rotationally equivalent, they can sit in quasi-equivalent positions. Two quasi-equivalent positions are the same shape and size but are not necessarily rotationally equivalent. Icosahedral structures can be made out of pentamers and hexamers, rather than just pentamers, which allows quasi-equivalent positions for the CPs. This is shown in Figure 2.1, where an icosahedral net is embedded into a hexagonal lattice, with the vertices of the icosahedron at the centres of hexagons. By choosing a different number of steps in each direction (h being horizontal and k being 60° above horizontal), a differently sized icosahedron with a different surface will be formed. These larger structures can be categorised by the following formula:

$$T(h,k) = h^2 + hk + k^2 \tag{2.25}$$

for integers h and k. This gives a virus its T-Number and viruses have $60T$ proteins making up their capsid. Looking at Figure 2.1d, there is a $T = 4$ capsid, shown to be made up of 80 triangles where each triangle indicates three protein positions, shown as dots. The edges of each triangle are the same as each other, contacting another triangle along their full length but the vertices are different, with some vertices having 5 triangles attached and some having 6. This is why these new positions are quasi-equivalent, rather than equivalent.

Due to these structures all exhibiting icosahedral symmetry, another feature of the icosahedron (or any symmetric object) can be applied to them: the fundamental domain (sometimes called the asymmetric unit). The entire surface of the icosahedron can be described by a small section of the surface, using the icosahedral group to reconstruct the entire surface from it. This small section of the surface that contains all the information to fully describe the capsid is called the Fundamental Domain (FD). FDs can be any shape that tessellates the entire icosahedral surface but kites are often used (as shown in the $T = 1$ icosahedron in Figure 2.1c).

However, as shown by Twarock and Luque [54], it does not need to be a hexagonal lattice that an icosahedral surface will be embedded into. Any lattice which contains a hexagonal sublattice can have an icosahedron embedded into it. Figure 2.2 shows the four lattices which can have an icosahedron embedded into them. Using this, viruses that form from more than one different protein (*e.g.* a major and a minor protein) can be classified. The figure also shows examples of convenient FDs for

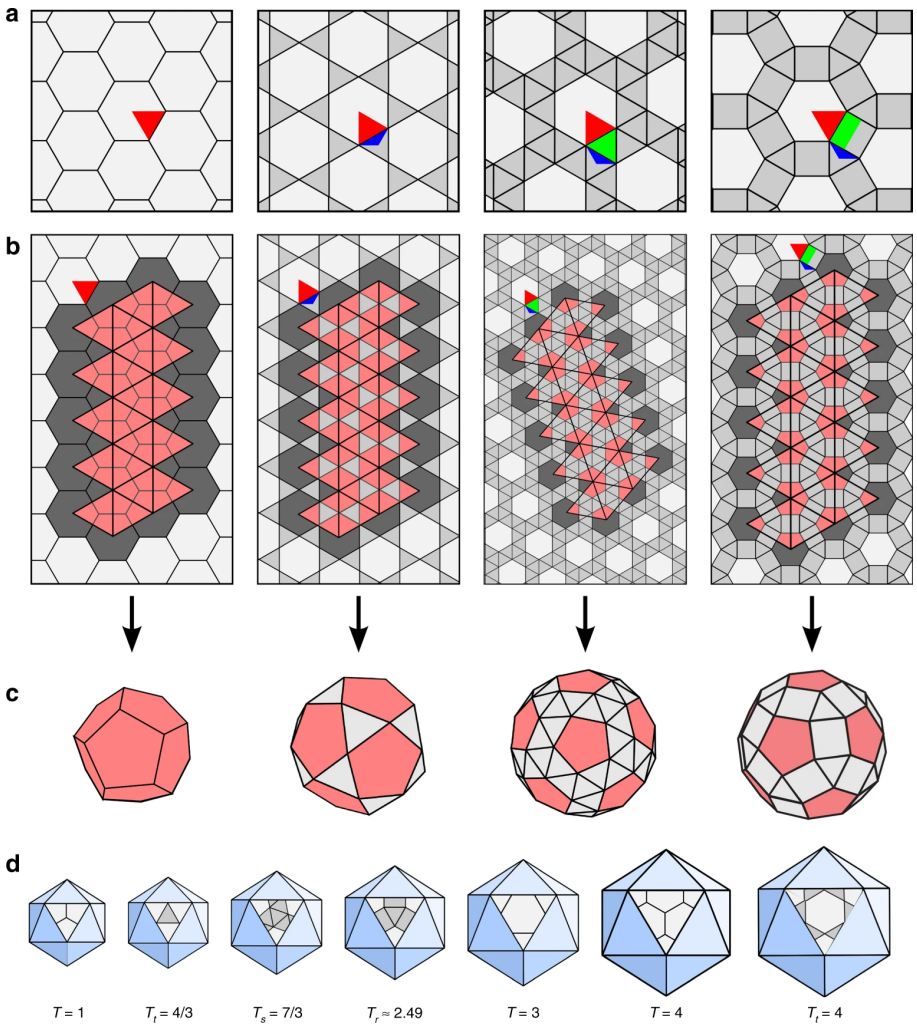

Figure 2.2: Figure and Caption adapted from [54]. Design of icosahedral architectures from Archimedean lattices. (a) The four Archimedean lattices permitting the Caspar-Klug construction (from left to right): the hexagonal, the trihexagonal, the snub hexagonal and the rhombitrihexagonal lattice. In each case, the asymmetric unit (repeat unit of the lattice) is highlighted. Its overlap with the hexagonal sublattice used for the construction of the icosahedral polyhedra is shown in red. Apart from the case of the hexagonal lattice, this also includes a third of a triangular surface (blue) and in addition a triangle or a half square (both shown in green) for two of the lattices, respectively. (b) Construction of Archimedean solids via replacement of 12 hexagons by pentagons in analogy to the Caspar-Klug construction (see also Figure 2.1b). (c) The polyhedral shapes corresponding to the examples shown in b. They each correspond to the smallest polyhedron in an infinite series of polyhedra for the given lattice type. (d) The smallest polyhedral shapes (T_t, T_s and T_r, denoting polyhedra derived from the trihexagonal, snub hexagonal and rhombitrihexagonal lattices, respectively) are shown organised according to their sizes in context with the Caspar-Klug polyhedra. As surface areas scale according to Equation 2.26 with respect to the Caspar-Klug geometries, the new solutions fall into the size gaps in between polyhedra in the Caspar-Klug series, or provide alternative layouts for capsids of the same size, as is the case for $T(2,0) = T_t(1,1) = \frac{4}{3}T(1,1) = 4$.

these embeddings. Recalling h and k and redefining them to be the number of steps from one hexagon to another, generalised T-Numbers can be defined as follows:

$$T_j(h,k) = \alpha_j(h^2 + hk + k^2) = \alpha_j T(h,k), \tag{2.26}$$

where $j = t, s, r$ represents the trihexagonal, snub hexagonal and rhombitrihexagonal lattices, respectively. A capsid labelled as $T_j(h,k)$ has the same number of hexagons and pentagons as one labelled $T(h,k)$ but also has additional shapes present. The variable α_j represents the scaling factor due to the difference in surface area that results, with $\alpha_t = \frac{4}{3}$, $\alpha_s = \frac{7}{3}$ and $\alpha_r = \frac{4}{3} + \frac{2}{\sqrt{3}}$ [54]. Further work has been done by Fatehi and Twarock [21], to represent capsid architecture by its interaction network. This approach, which is related to the tiling networks used here, enables wider classes of capsid architectures to be modelled, even if they cannot be easily represented as tilings. For example, this is the case if there are large holes in the capsid surface.

For icosahedral ssRNA viruses, the process that forms the capsid is actually a co-assembly process [13]. Both CPs and the RNA work together to form the capsid and ensure that the virus' RNA is specifically packaged into the capsid.

Viruses, their genetic information and their CPs are so small that they cannot be detected using visible light. To generate images of viruses, techniques such as Cryo-EM and X-Ray Crystallography must be used [33]. These involve firing either high energy electron beams or X-Rays at a sample of viruses and observing how the sample affects these beams. These techniques rely upon the icosahedral structure of viruses, as they use icosahedral averaging during image reconstruction. This means that if a capsid structure has a section which has asymmetric features (*e.g.* the RNA), then the data on these asymmetric features will be lost.

— 3 —

Models Of Equilibrium And Quasi-Equilibrium In Capsid Assembly

In this chapter, the existing work of Zlotnick [59] on the assembly of a dodecahedral capsid made of 12 pentameric capsomers in the absence of RNA is presented. Then, this method is extended, to allow for the presence of RNA in this model. The effect of both including the RNA and of changing various parameters are explored. After that, an alternate approach to the problem is developed. This approach alters the system of differential equations so they are all first-order. Numerical linear algebra methods are then applied to this system, so that the steady states can be identified and bifurcations they undertake can be found too.

In Caspar-Klug Theory [9], all viruses are made up of $60T$ proteins, where T is the T-Number of a virus. However, this leads to a sufficiently complex system so that it is not possible to analytically determine equilibrium dynamics for the system. For small ($T = 1$) viruses, made up of only 60 proteins, it is reasonable to model a system where the proteins form into pentamers, which then form a dodecahedral structure of twelve pentamers. In this chapter, two dodecahedral models of assembly, based on work done by Zlotnick [59] are discussed. These extend Zlotnik's model to include either RNA co-assembly or assembly where multiple intermediates can be present in the system. The first model looks into how the presence of RNA affects the k_D of viral formation, making it possible to predict a minimum ΔG_{prot} energy between proteins that enables stable capsid formation in RNA/Capsid co-assembly models. In the second model, quasi-equilibrium states and the effect of high ΔG_{prot} on kinetic trap formation are investigated.

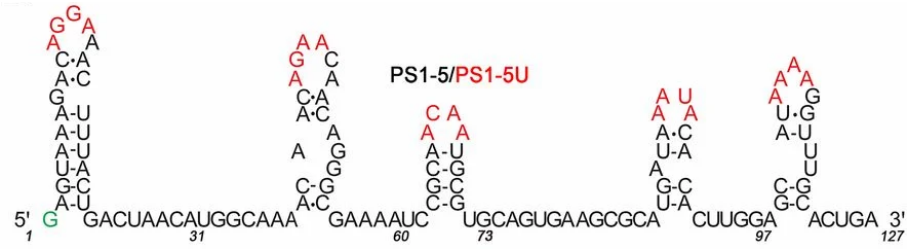

Figure 3.1: Figure *c.f.* from [42]. An example of an RNA sequence and the structures formed by its bases. These are believed to be the first five PSs for STNV, a $T = 1$ virus.

3.1 BACKGROUND TO DODECAHEDRAL MODELS

In the dodecahedral model of a $T = 1$ virus, first used by Zlotnick [59], 5 proteins come together to form pentagonal pentamers, which then bind to each other and form a dodecahedral capsid. Once there are 12 pentamers in the structure (60 proteins total), the capsid is complete. This is not how all viruses will form but it is reminiscent of some Picornaviruses, such as Rhinovirus and Poliovirus [59]. These viruses use a pentamer as the main sub-unit of the assembly, so pentamers will be referred to as capsomers, as this is the terminology for the smallest assembly unit. In the original work by Zlotnick, the effects of RNA on the assembly are not considered.

It is important to consider that the RNA may be present during assembly, as it needs to be packaged inside of the capsid. As will be shown, it can have additional energetic effects on assembly and k_D.

RNA is a sequence made of four bases, A, G, C and U, which are able to pair with each other according to Watson-Crick base pairing. Viral RNAs form structures, which can bind to proteins/protein structures (the pentamers here), termed PSs. An example of an RNA sequence and the structures it forms are shown in Figure 3.1. These are usually understood to assist with the assembly of the capsid and help ensure that the RNA is packaged inside the capsid.

Following Zlotnick, we assume that the capsomers will be added into the structure at the position that creates the largest number of capsomer-capsomer bonds, to ensure we follow only the lowest energy pathway. This assumption can be backed up by Figure 7.20 in Section 7.2.4, with stochastic simulations of dodecahedral assembly having over 80% of intermediates of any size (over 95% for many sizes of intermediate) as the most stable intermediate available. Some of the lowest energy intermediates (and some examples of higher energy intermediates) are shown in Figure 3.2. We

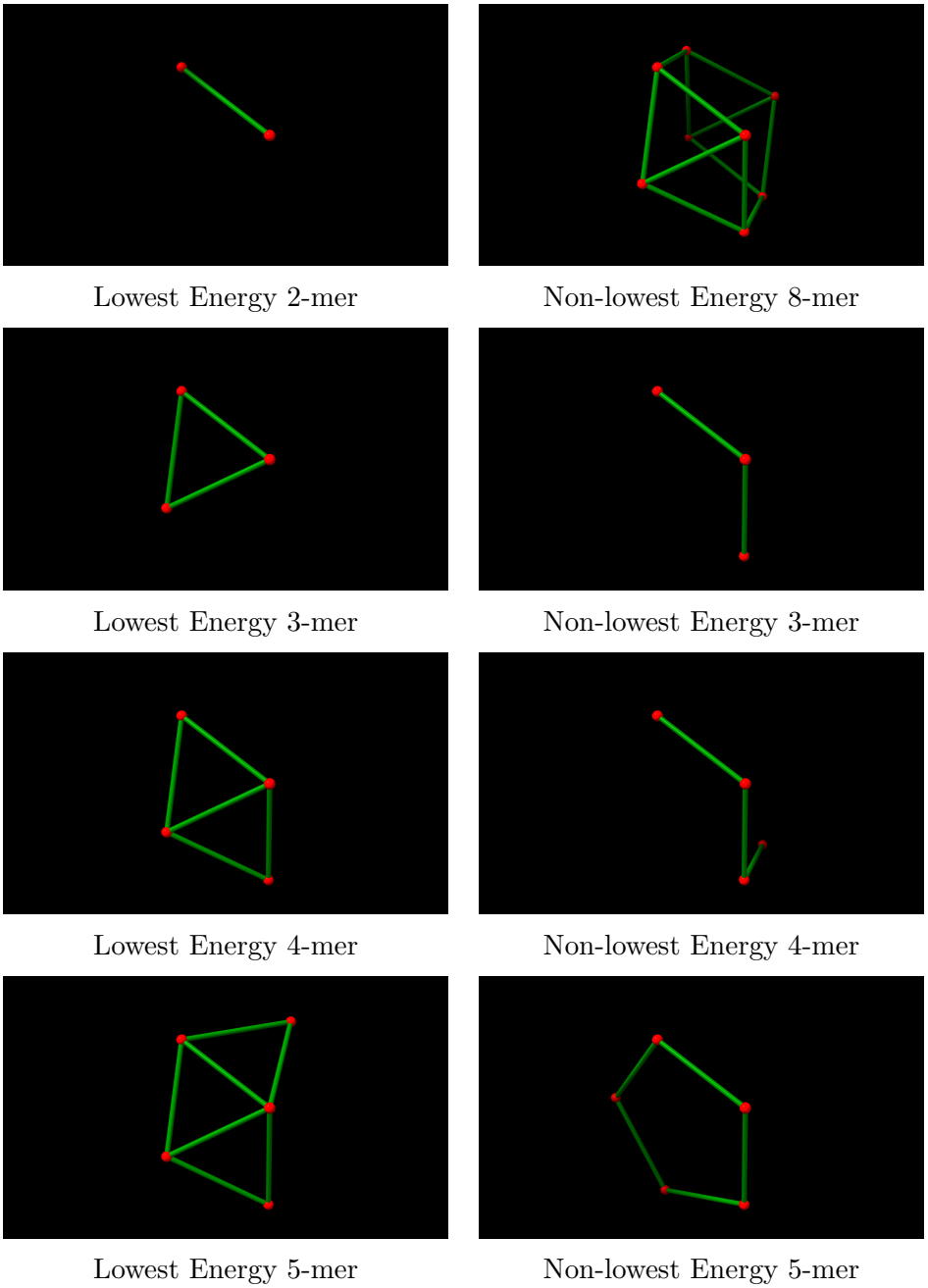

Figure 3.2: Examples of both the lowest energy intermediates and some non-lowest energy intermediates from the assembly of a dodecahedral capsid. Each dot represents a capsomer and each line represents a contact between adjacent protein capsomers.

also neglect modelling of pentamer formation as we assume the assembly pathway consists of pentamer capsomers combining to form larger oligomers and also assume that the rate of the formation of the pentamer is fast, relative to the rate of formation of the capsid. These above assumptions are used throughout this chapter.

Before going into detail of the reactions, we need to clarify some notation. We define (i) to be an intermediate with i capsomers. This will be the lowest energy intermediate of that size, as we are assuming that only the lowest energy intermediates are formed. The concentration of the intermediate with i capsomers is written as $[i]$ and can be measured as either a molar concentration or as the number of particles in a volume.

We only consider single capsomer addition and removal in each reaction. The forward reaction, often known as an elongation reaction, is where an extra capsomer binds to the existing intermediate structure of one or more capsomers, increasing its size. This can be written as

$$(1) + (j) \rightleftharpoons (j+1), \tag{3.1}$$

where (j) is the intermediate with j capsomers. The backwards reaction, also known as a removal reaction, is where a capsomer leaves the existing protein structure, decreasing its size. Both reactions are allowed as all of these assembly reactions are considered reversible reactions.

The rate equation for the concentration of an intermediate formed from j capsomers is:

$$\frac{d[j]}{dt} = k_f^j[1][j-1] + k_b^{j+1}[j+1] - k_f^{j+1}[1][j] - k_b^j[j], \tag{3.2}$$

where k_f^j is the forward reaction rate for an intermediate of size $j-1$ gaining a capsomer, k_b^j is the backwards reaction rate for an intermediate of size j losing a capsomer and $[j]$ is the concentration of an intermediate of size j.

The values of k_f^j and k_b^j can be set as follows, using the statistical factors, $S_U^P(j)$ and $S_D^P(j)$ given in Table 3.1. These statistical factors arise due to there being multiple equivalent ways to add or remove a capsomer and result in the same intermediate formed from a reaction.

$$k_f^j = 5 \cdot S_U^P(j) \cdot k \tag{3.3}$$

$$k_b^j = S_D^P(j) \cdot k e^{\beta \Delta G_{protC}(j)}, \tag{3.4}$$

n	$S_U^R(n)$	$S_D^R(n)$	$S_U^P(n)$	$S_D^P(n)$	c(n)
1	-	-	-	-	-
2	5	1	5/2	1	1
3	2	1	2	3	2
4	2	1	3	2	2
5	2	1	4	2	2
6	1	1	1	5	3
7	2	1	5	1	2
8	2	1	2	4	3
9	1	1	2	3	3
10	2	1	3	2	3
11	2	1	2	5	4
12	1	1	1	12	5

Table 3.1: A table of the statistical factors in dodecahedral assembly, formed from 12 pentameric capsomers, assuming it follows the path of lowest energy. The variable n is the number of capsomers present in the structure. $S_U^R(n)$ and $S_U^P(n)$ are the number of ways that a capsomer can be added to create the most stable intermediate with n capsomers when the RNA is present and absent respectively. $S_D^R(n)$ and $S_D^P(n)$ are the number of ways that a capsomer can be removed to create the most stable intermediate with $n-1$ capsomers when the RNA is present and absent, respectively. $S_D^R(n)$ is always 1, as only the capsomer at the loose end of the RNA is allowed to leave the structure. The column $c(n)$ represents the number of capsomer-capsomer contacts formed when going from the structure with $n-1$ capsomers to the structure with n capsomers. The columns with RNA absent are from Zlotnick [59] and the columns for RNA present were calculated by hand. Some structures are shown in graph form in Figure 3.2.

where k is a constant which is independent of j, $\beta = \frac{1}{RT}$ and $c(j)$ is the number of capsomer-capsomer contacts formed when going from the structure with $j-1$ capsomers to the structure with j capsomers.

To generate the statistical factors, we look to Figure 3.3. The figure shows the possible locations that a new capsomer could join the structure, whilst forming the maximal number of bonds, *i.e.* following the minimal energy pathway for assembly. This is only done here for intermediates with 1 (A), 2 (B) or 3 (C)/(D) capsomers present but can be done for larger intermediates, too. This allows us to calculate $S_U^P(j)$ and $S_U^R(j)$, the number of ways that a capsomer can be added to create the most stable intermediate with j capsomers when the RNA is absent and present respectively. It also allows us to calculate $S_D^P(j)$, the number of ways that a capsomer can be removed to create the most stable intermediate with $j-1$ capsomers when

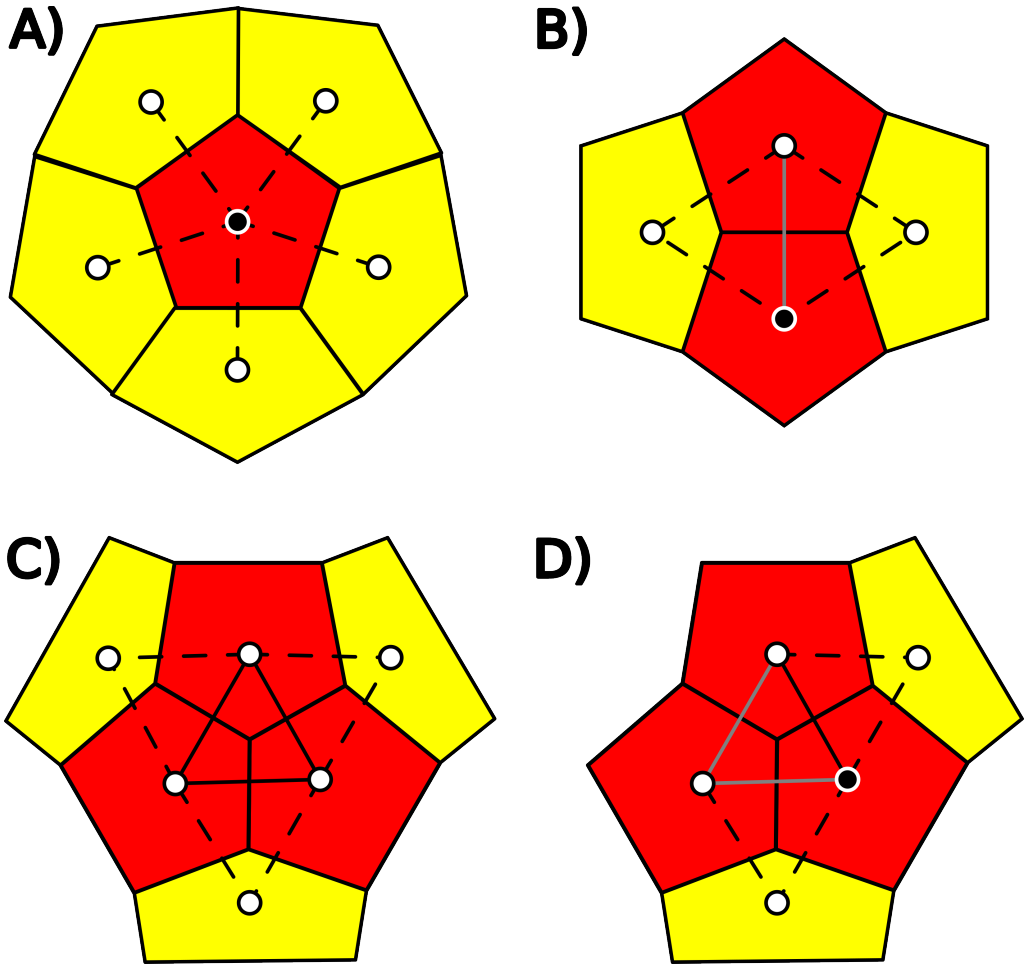

Figure 3.3: A planar demonstration of where some of the statistical factors in Table 3.1 come from. Each pentagon represents an identical capsomer from the dodecahedral assembly model, with red pentagons representing capsomers already present in the structure and yellow indicating positions new capsomers could join the structure whilst still following the lowest energy path. Within each pentagon is a circle that represents the capsomer as a node and the circle with the white border represents (when the RNA is present) the current position of the PS adjacent to the loose RNA. When the RNA is present, the assembly is restricted so that any capsomer added must be adjacent to the previously added capsomer, so the RNA traces out a path on the interior of the capsid. Solid lines indicate existing bonds and dotted lines indicate potential bonds. In the absence of RNA, structures are considered to be equivalent when one is rotationally equivalent to another. When the RNA is present, the structures are considered equivalent when structures are rotationally equivalent, with both free ends of the RNA at the same position. (A) and (B) represent an intermediate made of 1 or 2 capsomers, respectively and are equivalent whether the RNA is present or absent. (C) and (D) show the differences for a intermediate of 3 capsomers based on whether RNA is absent or present, respectively.

the RNA is absent. During Section 3.2, we will assume that the RNA starts assembly at one end and only adds/removes proteins at this loose end of the RNA. As a result, $S_D^R(j) = 1$ for all j, as it will only ever remove the most recently added protein.

In (A), we see that there are 5 different ways to add a capsomer to the one present (due to the 5-fold rotational symmetry). Due to degeneracy of the structure formed from two pentameric capsomers, this gives a value of $S_U^P(2) = \frac{5}{2}$ and $S_U^R(2) = \frac{5}{2}$. In (B), we have two capsomers and two different ways to add a new capsomer whilst still forming two bonds, which gives $S_U^P(3) = 2$ and $S_U^R(3) = 2$. There is only one way to break a bond here, so this gives $S_D^P(2) = 1$. In (C) and (D), we have the first time when S_U^P and S_U^R differ. For (C), we have three capsomers and there are three different ways to add a new capsomer and form two bonds, giving $S_U^P(4) = 3$. We also have three ways to remove a capsomer and get a minimum energy dimer, so $S_D^P(3) = 3$. For (D), one of the capsomers is not accessible to the free end of the RNA, so $S_U^R(4) = 2$. A full list of these statistical factors is in Table 3.1.

These reactions are covered, for the case where RNA is present, in more detail in the next section.

3.2 APPARENT RATE OF CAPSOMER-CAPSOMER BINDING IN AN RNA ASSEMBLY MODEL

Zlotnicks rates k_f^i are capsomer-capsomer binding rates in the RNA-free case. When RNA is present, there is an additional reaction for the RNA-capsomer binding, as can be seen in Figure 3.4. However, instead of using 23 reactions: 11 from Zlotnick and 12 of RNA-capsomer binding, we encompass the RNA binding and the capsomer-capsomer binding into a single apparent forward and backwards rate: $k_f^{app}(j)$ and $k_b^{app}(j)$. The one unusual reaction is where $j = 1$, which just represents a single capsomer being added to a PS on the RNA, with no capsomer-capsomer binding occurring. Using these rates allows the comparison of the RNA case with the RNA-free case directly within the Zlotnick Model. Markov Chains are used to analyse these reactions and to generate these apparent rates. The aim is to calculate what values of ΔG_{prot} and ΔG_{rna} are needed for efficient assembly and how this changes between cases.

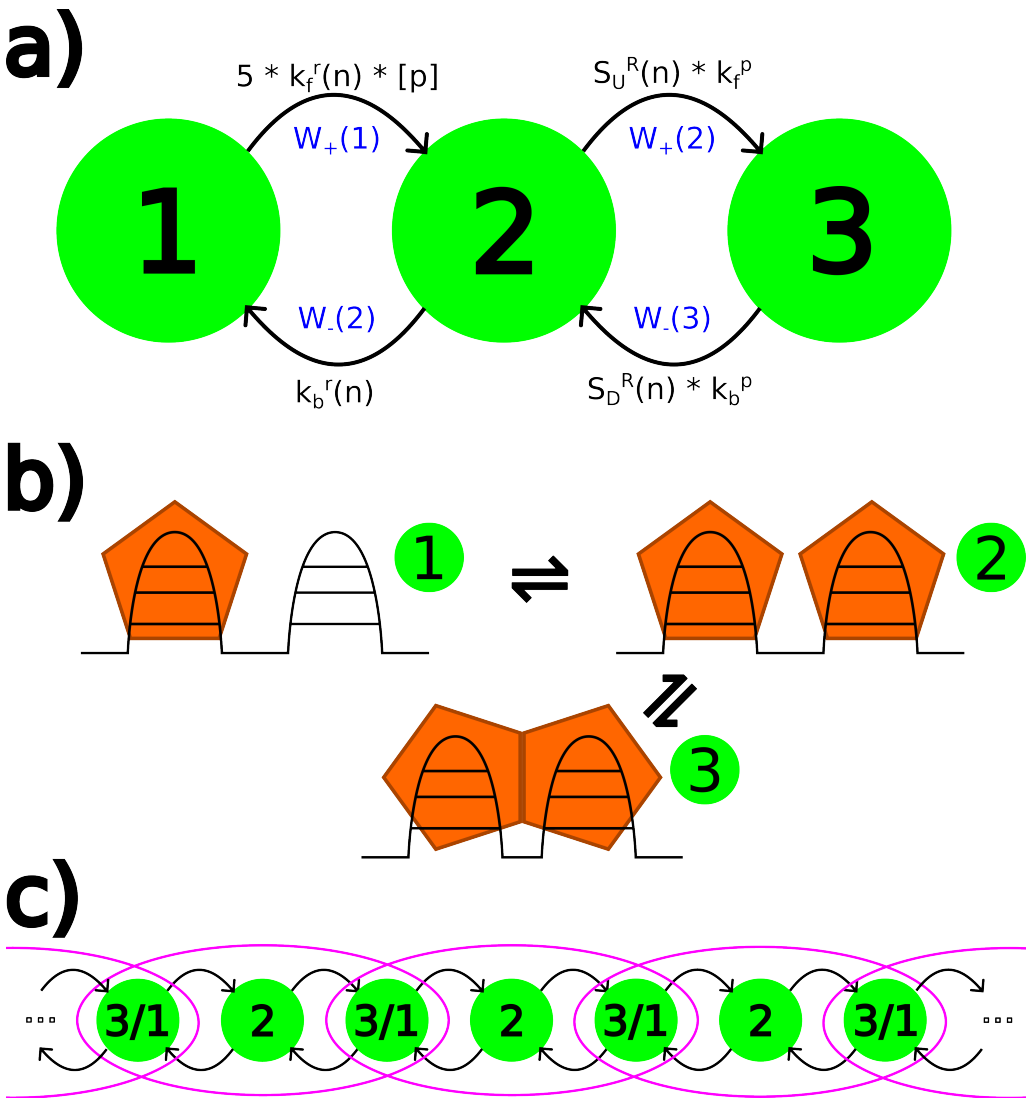

Figure 3.4: Figure of the states involved in the assembly process including RNA. The states labelled as (1), (2) or (3) are equivalent in this figure. (a) Shows the transitions between states that can occur and their relative reaction rates ($W_{\pm}(m)$ for the states $m = 1, 2, 3$). (b) The states present in each step of the assembly process. In each state, any PSs to the left of the displayed RNA section are also bound, their capsomers make up part of the existing structure and this is connected to the left PS's capsomer. Any PSs to the right of the RNA section shown are unbound and free. In state 1, there is a capsomer bound to the left PS and an unbound PS adjacent on the displayed section of the RNA. In state 2, both PSs shown are bound to a capsomer. In state 3, the capsomers bound to the two PSs have now bound to each other as well, increasing the size of the assembly intermediate. (c) Shows how the states in (a) overlap and interact to form (part of) the full chain which describes assembly from just an RNA to a complete capsid.

3.2.1 $K_{D^{app}}$ In The Absence Of RNA

In Zlotnick's paper, a quantity called $K_{D^{app}}$ was determined. This represents the concentration of proteins needed for the system to have an equal concentration of protein and capsid, *i.e.* $K_{D^{app}} = [1]$ and also $K_{D^{app}} = [12]$. It was calculated analytically as shown below.

First, we consider a system that has been left for long enough that it has reached chemical equilibrium. The system will only include 12 intermediates and (other than free capsomers and completed capsids, which only have one) each intermediate has exactly two reversible reactions that can create an intermediate of size n. One is an addition reaction to an intermediate of size $n-1$ and one is a removal reaction from an intermediate of size $n+1$. This means that for the system to be in equilibrium, the forward and backwards rates for every reaction must be equal, so the net rate of reaction is zero. So, for each reaction, $(n-1)+(1) \rightleftharpoons (n)$, at equilibrium we have

$$k_b^n[n] = k_f^n[1][n-1], \tag{3.5}$$

where k_f^n is the forward reaction rate for an intermediate of size $n-1$ gaining a capsomer, k_b^n is the backwards reaction rate for an intermediate of size n losing a capsomer. Rearranging this gives

$$[n] = \frac{k_f^n}{k_b^n}[n-1][1]. \tag{3.6}$$

Applying this recursively, we get

$$[n] = [1]^n \prod_{i=2}^{n} \frac{k_f^i}{k_b^i}. \tag{3.7}$$

Inserting $n = 12$, we get an equation for the concentration of completed capsid:

$$[12] = [1]^{12} \prod_{i=2}^{12} \frac{k_f^i}{k_b^i}. \tag{3.8}$$

For the RNA-free scenario, we can recall from Equation 3.4:

$$\frac{k_f^i}{k_b^i} = 5 \cdot \frac{S_U^P(i)}{S_D^P(i)} e^{-\beta \cdot c(i) \cdot \Delta G_{prot}}. \tag{3.9}$$

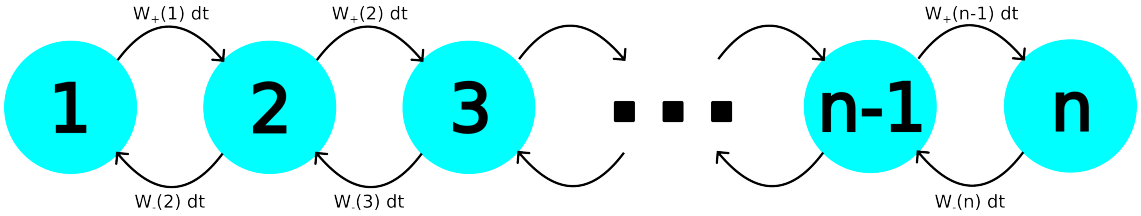

Figure 3.5: Demonstration of the probabilities of transition between states in a Markov Chain with n states, where state n is absorbing. With $W_{\pm}(i)$, which is defined in Equation 3.13, where $W_{\pm}(i) \cdot dt$ is the probability of transitioning from state i to state $i \pm 1$ before time $t + dt$.

We simplify this to

$$\begin{aligned}\prod_{i=2}^{12} \frac{k_f^i}{k_b^i} &= \prod_{i=2}^{12} 5 \cdot \frac{S_U^P(i)}{S_D^P(i)} e^{-\beta \cdot c(i) \cdot \Delta G_{prot}} \\ &= 5^{11} \cdot e^{-\beta \Delta G_{prot} \sum_{i=2}^{12} c(i)} \cdot \prod_{i=2}^{12} \frac{S_U^P(i)}{S_D^P(i)} \\ &= 5^{11} \cdot e^{-30\beta \Delta G_{prot}} \cdot \frac{1}{60} \\ &= \frac{5^{11}}{12} \cdot e^{-30\beta \Delta G_{prot}},\end{aligned}$$

which altogether gives

$$\frac{[12]}{[1]^{12}} = \frac{5^{11}}{12} \cdot e^{-30\beta \Delta G_{prot}}. \tag{3.10}$$

To find $K_{D^{app}}$, we set $K_{D^{app}} = [1] = [12]$ and rearrange to give

$$K_{D^{app}} = \sqrt[11]{\frac{12}{5^{11}} \cdot e^{30\beta \Delta G_{prot}}}. \tag{3.11}$$

At a temperature of $T = 298K$ and $\Delta G_{prot} = -4.08$, this gives $K_{D^{app}} \approx 1.729$nM.

3.2.2 Mean First Passage Time

To construct an apparent capsomer-capsomer binding rate in the presence of RNA, we need to calculate the mean first passage time for reaction j: τ_j and estimate $k_f^{app}(j) = \frac{1}{\tau_j}$.

A few definitions are needed. In Markov Chains, there are a number of discrete states whose future depends only on the present. The time in this system is treated as continuous. The system can only change by jumping either up or down a state. These jumps each have a probability associated with them. Figure 3.5 shows a

graphical illustration of Markov Chains. We have $a(n)dt$, which is the probability, at time t, that state n will jump away before time $t + dt$. It can also be thought of as the flux or rate for this reaction. We also have $w_{\pm}(n)$, the probability that a process jumping from state n at time t will land in state $n \pm 1$. As it can only jump up or down, we get:

$$w_{+}(n) + w_{-}(n) = 1. \tag{3.12}$$

Lastly, we define $W_{\pm}(n)dt$ as the probability that a process at state n at time t will jump to state $n \pm 1$ before time $t + dt$. These three probabilities are connected by:

$$W_{\pm}(n) = a(n)w_{\pm}(n). \tag{3.13}$$

It follows from Equation 3.12 that

$$W_{+}(n) + W_{-}(n) = a(n). \tag{3.14}$$

As they are all probabilities, we can say that for all states n:

$$a(n) \geq 0 \tag{3.15}$$

$$w_{\pm}(n) \geq 0 \tag{3.16}$$

$$W_{\pm}(n) \geq 0. \tag{3.17}$$

Taken together with Equation 3.12, this leads to:

$$-a(n) \leq W_{+}(n) - W_{-}(n) \leq a(n). \tag{3.18}$$

To find the mean first passage time to go from state $n_0 \to n_1$, you find the total time before the first time that reaction completes and the inverse of that is the apparent rate of the equation. When going from state n_0 to n_1, the average time spent in the $n_1 - 1$ state and the average time spent in another state n are given by:

$$t(n_1 - 1; n_0 \to n_1) = \frac{1}{W_{+}(n_1 - 1)} \tag{3.19}$$

$$t(n; n_0 \to n_1) = \frac{\theta(n + 1 - n_0)}{W_{+}(n)} + \frac{W_{-}(n+1)}{W_{+}(n)} t(n + 1; n_0 \to n_1), \tag{3.20}$$

where we define the unit step function:

$$\theta(x) = \begin{cases} 0, \text{ if } x \leq 0 \\ 1, \text{ if } x > 0 \end{cases} \tag{3.21}$$

Thus, the total time to transfer from state n_0 to n_1 is:

$$T(n_0 \to n_1) = \sum_{n=0}^{n_1 - 1} t(n; n_0 \to n_1). \tag{3.22}$$

3.2.3 CALCULATION OF $k_f^{app}(j)$ AND $k_b^{app}(j)$ USING MFP TIME

We are interested in finding k_f and k_b for the addition of each protein, for systems where the RNA is both present and absent. The RNA makes this a two-step process, so to get the overall rate is a more complicated calculation These rates allow us to compare the rates of the process when RNA is present or absent. We use mean first passage for this, to approximate the average time it takes for the system to go from one state to another one and take the inverse to get the rate.

In this section, we consider the RNA as well as the capsomers. As a result, the notation will shift slightly, so [1] will no longer be the concentration of free capsomers, but the concentration of intermediates made up by a single capsomer bound to an RNA. The notation for the concentration of free capsomers will be $[p]$ and the concentration of the RNA will be $[r]$.

The system begins as just an RNA, with no capsomers bound to it. The primary reaction is simple to model as it requires no capsomer-capsomer bond once the capsomer-PS bond has formed. The rates can be written as:

$$W_-(1) = 0 \tag{3.23}$$
$$W_+(1) = 5k_f^r(1)[p] \tag{3.24}$$
$$W_-(2) = k_b^r(1) \tag{3.25}$$
$$W_+(2) = 0, \tag{3.26}$$

where $[p]$ is the concentration of pentameric capsomers in the mixture, $k_f^r(1)$ and $k_b^r(1)$ are the on and off rate, respectively, for the 1st PS. In this case, $T(1 \to 2) = t(1; 1 \to 2) = \frac{1}{5k_f^r(1)[p]}$. This means we have $k_f^{app}(1) = 5k_f^r(1)$ for the first step of the reaction as $k_f^{app} = \frac{1}{T[p]}$ in reactions that depend on concentration of free capsomers and $k_b^{app} = k_b^r(1)$ as $k_b^{app} = \frac{1}{T}$ when the reaction does not depend on the concentration of free capsomers.

For further reactions, this can be made more general. As shown in Figure 3.4, there are 3 states that are present in the reactions to add a new capsomer. There is state 1, where there is a structure made of $n-1$ capsomers and a free PS adjacent to the structure. State 2 has the same structure of $n-1$ capsomers but a new capsomer has bound to the PS adjacent to the structure. State 3 is where the new capsomer has joined the structure, so it is now made of n capsomers. We write, for a structure

of n capsomers:

$$W_-(1) = 0 \tag{3.27}$$
$$W_+(1) = 5k_f^r(n)[p] \tag{3.28}$$
$$W_-(2) = k_b^r(n) \tag{3.29}$$
$$W_+(2) = S_U^R(n)k_f^p \tag{3.30}$$
$$W_-(3) = S_D^R(n)k_b^p \tag{3.31}$$
$$W_+(3) = 0. \tag{3.32}$$

Where $S_U^R(n)$ and $S_D^R(n)$ are statistical factors for the up and down reactions, given in Table 3.1, due to the symmetry of the dodecahedron. k_f^p and k_b^p are the on and off rates for the capsomer bound to the next PS along the RNA to bind to the structure, respectively. These probabilities give us:

$$t(2; 1 \to 3) = \frac{1}{S_U^R(n)k_f^p} \tag{3.33}$$
$$t(1; 1 \to 3) = \frac{1}{5k_f^r(n)[p]} + \frac{k_b^r(n)}{5k_f^r(n)[p]}\frac{1}{S_U^R(n)k_f^p}. \tag{3.34}$$

Taking the total of these, we get

$$\begin{aligned} T(1 \to 3) &= \frac{1}{S_U^R(n)k_f^p} + \frac{1}{5k_f^r(n)[p]} + \frac{k_b^r(n)}{5k_f^r(n)[p]}\frac{1}{S_U^R(n)k_f^p} \\ &= \frac{5k_f^r(n)[p] + k_b^r(n) + S_U^R(n)k_f^p}{5k_f^r(n)[p]S_U^R(n)k_f^p}, \end{aligned}$$

which leads to:

$$k_f^{app}(n) = \frac{5S_U^R(n)k_f^p k_f^r(n)}{5k_f^r(n)[p] + k_b^r(n) + S_U^R(n)k_f^p} \tag{3.35}$$

being the apparent forward reaction rate that leads to a structure of n capsomers. We can follow similar working for the reverse:

$$t(2; 3 \to 1) = \frac{1}{k_b^r(n)} \tag{3.36}$$
$$t(3; 3 \to 1) = \frac{1}{S_D^R(n)k_b^p} + \frac{S_U^R(n)k_f^p}{S_D^R(n)k_b^p} \cdot \frac{1}{k_b^r(n)}. \tag{3.37}$$

These sum to give

$$\begin{aligned} T(3 \to 1) &= \frac{1}{k_b^r(n)} + \frac{1}{S_D^R(n)k_b^p} + \frac{S_U^R(n)k_f^p}{S_D^R(n)k_b^p} \cdot \frac{1}{k_b^r(n)} \\ &= \frac{S_D^R(n)k_b^p + k_b^r(n) + S_U^R(n)k_f^p}{k_b^r(n)S_D^R(n)k_b^p}, \end{aligned}$$

$$k_b^{app}(n) = \frac{k_b^r(n) S_D^R(n) k_b^p}{S_D^R(n) k_b^p + k_b^r(n) + S_U^R(n) k_f^p} \tag{3.38}$$

being the apparent rate at which a structure of n capsomers decays to a structure of $n-1$ capsomers. We now take the following ratio:

$$\frac{k_f^{app}(n)}{k_b^{app}(n)} = \frac{5 S_U^R(n) k_f^p k_f^r(n)}{S_D^R(n) k_b^p k_b^r(n)} \frac{S_D^R(n) k_b^p + k_b^r(n) + S_U^R(n) k_f^p}{5 k_b^r(n)[p] + k_b^r(n) + S_U^R(n) k_f^p}, \tag{3.39}$$

which, using $k_f^p / k_b^p = e^{-\beta c(n) \Delta G_{prot}}$ and $k_f^r(n) / k_b^r(n) = e^{-\beta \Delta G_{rna}(n)}$, where $\beta = \frac{1}{RT}$, simplifies to:

$$\frac{k_f^{app}(n)}{k_b^{app}(n)} = 5 \frac{S_U^R(n)}{S_D^R(n)} e^{-\beta c(n) \Delta G_{prot}} \cdot e^{-\beta \Delta G_{rna}(n)} \cdot f(n, [p]). \tag{3.40}$$

Here

$$f(n, [p]) = \frac{S_D^R(n) k_b^p + k_b^r(n) + S_U^R(n) k_f^p}{5 k_b^r(n)[p] + k_b^r(n) + S_U^R(n) k_f^p}. \tag{3.41}$$

Comparing this to the equivalent from Zlotnick, where the RNA was not considered, gives us:

$$\frac{k_f(n)}{k_b(n)} = 5 \frac{S_U^P(n)}{S_D^P(n)} e^{-\beta c(n) \Delta G_{prot}}, \tag{3.42}$$

which is similar but the RNA version has two extra terms to be multiplied in, due to the RNA's presence. One is simply the inclusion of the capsomer-PS bond energy. However, $f(n, [p])$ is a term that shows the RNA's presence does more than simply lower the necessary energy for the structure to form by the same amount as its bond strength. It makes it necessary to model the removal of a capsomer as a two-step process. It therefore reduces the speed at which this can happen, making the capsid intermediates more stable.

Including RNA in the assembly model imposes restrictions onto the assembly process. In the absence of RNA, a new capsomer is able to join at any edge of the intermediate (assuming it is on the lowest energy pathway), whereas in the presence of RNA, a new capsomer must join adjacent to the capsomer that joined immediately before it. This restriction means that the values of $S_U(n)$ and $S_D(n)$ are different when the RNA is present compared to when it is not. All values can be seen in Table 3.1, for both RNA present and RNA absent.

Considering $f(n, [p])$, it has three major components:

$$A = S_D^R(n) k_b^p \tag{3.43}$$

$$B = k_b^r(n) + S_U^R(n) k_f^p \tag{3.44}$$

$$C = 5 k_b^r(n)[p]. \tag{3.45}$$

Therefore, you can write

$$f(n,[p]) = \frac{A+B}{C+B}. \tag{3.46}$$

The ratios of these three values determine the change caused by this term, due to the presence of RNA. If $A >> B$ and $C << B$ then the forward reaction will be boosted by the presence of the RNA. If $A << B$ and $C >> B$ then the backwards reaction will be stronger due to the presence of the RNA. If both $A << B$ and $C << B$, or if $A \approx C$, then $f(n,[p]) \approx 1$, so the reaction will not be affected.

3.2.4 Calculating $K_{D^{app}}$ In The Presence Of RNA

In assemblies that include that RNA, before capsomers can join the capsid structure, they must first bind to a PS and are then able to add to the structure. It is assumed that the on/off reactions for the capsomer-PS bonds are much faster than the on/off reactions for the capsomer-capsomer bonds, so that RNA-capsomer contacts equilibrate prior to capsomer-capsomer contacts. Experimental work has observed that in many viruses, the RNA-Protein interactions are not part of the rate limiting step [23], which supports this assumption. Additionally, as this system only considers one RNA and an infected cell would contain a large number of RNAs, proteins and capsomers, it is also assumed that the RNA binding to capsomers has a negligible effect on the concentration of capsomers, which is therefore assumed constant.

Equation 3.40 allows us to estimate the value of $K_{D^{app}}$ when the RNA is present. This is done by defining $K_{D^{app}} = [p] = [12] = [r]$ as the equilibrium concentration where all reactants are present in equal concentrations. Then, we use the equilibrium equation for the first reaction:

$$[1] = \frac{k_f^{app}(1)}{k_b^{app}(1)}[r][p] \tag{3.47}$$

and the general form as before:

$$[n] = \frac{k_f^{app}(n)}{k_b^{app}(n)}[n-1][p]. \tag{3.48}$$

Applying this recursively gives

$$[12] = \left(\prod_{i=1}^{12} \frac{k_f^{app}(n)}{k_b^{app}(n)}\right)[p]^{12}[r]. \tag{3.49}$$

		ΔG_{prot}				
		0.0	-2.0	-4.0	-6.0	-8.0
ΔG_{rna}	0.0	1.1	1.2×10^{-4}	2.7×10^{-8}	5.9×10^{-12}	1.3×10^{-15}
	-2.0	2.0×10^{-2}	4.3×10^{-6}	9.3×10^{-10}	2.0×10^{-13}	4.4×10^{-17}
	-4.0	4.6×10^{-4}	1.5×10^{-7}	3.2×10^{-11}	6.9×10^{-15}	1.5×10^{-18}
	-6.0	1.5×10^{-5}	5.0×10^{-9}	1.1×10^{-12}	2.4×10^{-16}	5.1×10^{-20}
	-8.0	5.2×10^{-7}	1.7×10^{-10}	3.7×10^{-14}	8.1×10^{-18}	1.7×10^{-21}
RNA-free		2.5×10^{-1}	2.5×10^{-5}	2.5×10^{-9}	2.5×10^{-13}	2.5×10^{-17}

Table 3.2: A table containing the values of $K_{D^{app}}$ for dodecahedral systems that include RNA with a variety of uniform PS-capsomer bond affinities (in kcal/mol) and a variety of capsomer-capsomer bonding affinities (in kcal/mol), that can be compared to the exact values of $K_{D^{app}}$ calculated using the formula in Equation 3.11 for the RNA-free case. This represents the equilibrium concentration where $K_{D^{app}} = [p] = [12] = [r]$. The row for $\Delta G_{rna} = 0$ models a system where RNA is present but the bonds it forms are negligible, to show the effect of the RNA's presence on the assembly. The values of $K_{D^{app}}$ are in mol/dm^3.

Code was written that would generate $K_{D^{app}}$ to 1 decimal place, for given ΔG_{prot} and ΔG_{rna} values, such that

$$K_{D^{app}} - \left(\prod_{i=1}^{12} \frac{k_f^{app}(n)}{k_b^{app}(n)} \right) K_{D^{app}}^{13} = 0. \tag{3.50}$$

The results of this code is stored in Table 3.2 and the values of

$$\Delta G_{app} = RT \ln(K_{D^{app}}) \tag{3.51}$$

are stored in Table 3.3 for comparisons to the exact calculations from the equation:

$$\Delta G_{app} = \frac{30\Delta G_{prot} - RT \ln(\frac{5^{11}}{12})}{11} \tag{3.52}$$

for the RNA-free case.

Considering Table 3.2, the concentrations in each column are always close to the concentration provided by Zlotnick's formula. However, they are usually at slightly lower concentrations which suggests that the presence of the RNA gives a stabilising effect, which helps lower the concentration of capsomers needed for $[p] = [12]$.

Looking more closely at Table 3.3, more information can be found. The first row gives information on what the ΔG_{app} value is when the capsid forms alongside an

dGapp		dGprot				
		0.0	-2.0	-4.0	-6.0	-8.0
dGRNA	0.0	0.16	-15.24	-29.41	-43.64	-57.85
	-2.0	-6.60	-20.86	-35.10	-49.35	-63.57
	-4.0	-12.97	-26.52	-40.79	-55.03	-69.27
	-6.0	-18.75	-32.26	-46.47	-60.70	-74.98
	-8.0	-24.42	-37.97	-52.20	-66.42	-80.72
RNA-free		-2.34	-17.89	-33.43	-48.98	-64.53

Table 3.3: A colour-coded table, containing the ΔG_{app} values for a range of ΔG_{rna} and ΔG_{prot} values. In the colour scheme, red indicates the largest values and green the smallest, with all others on a spectrum between. These values were calculated placing values from Table 3.2 into Equation 3.51. The final row represents the exact calculation given by Zlotnick for a dodecahedral capsid in the absence of RNA. All ΔG values are given in kcal/mol.

RNA that makes negligible bonds to the capsomers. These values are always less stable than the values for the RNA-free row, suggesting that assembly featuring the RNA brings extra constraints with it, that limit and reduce the effectiveness of the assembly. However, this is normally cancelled out by the extra stabilising energy that the RNA provides via its PS-capsomer bonds. On other rows, where the PS-capsomer bonds are non-negligible, the ΔG_{app} values are (with the exception of the $\Delta G_{prot} = -8.0$ and $\Delta G_{rna} = -2.0$ cases) more stable than the RNA-free case. The larger the values of ΔG_{prot} and ΔG_{rna}, the larger the resultant values of ΔG_{app}. Increasing ΔG_{prot} had more effect on increasing ΔG_{app} than increasing ΔG_{rna}.

One conclusion of the above calculations is that as you increase one of the two bond strengths, the amount of completed capsids present in a system would increase. In Figure 3.6, there is a graph that shows the yield from the assembly of a dodecahedral capsid, with two different RNAs, with different ΔG_{rna} values and one RNA-free assembly, where only the capsomer-capsomer binding energy is being altered. This simulation uses the Gillespie algorithm, covered in Section 2.2, to model a system with enough capsomers to generate 2000 completed capsids based on reactions described above. The model used for this figure allows intermediates other than the lowest energy intermediates (though, as shown in Figure 7.20, most intermediates that form during the assembly are the lowest energy states). It shows

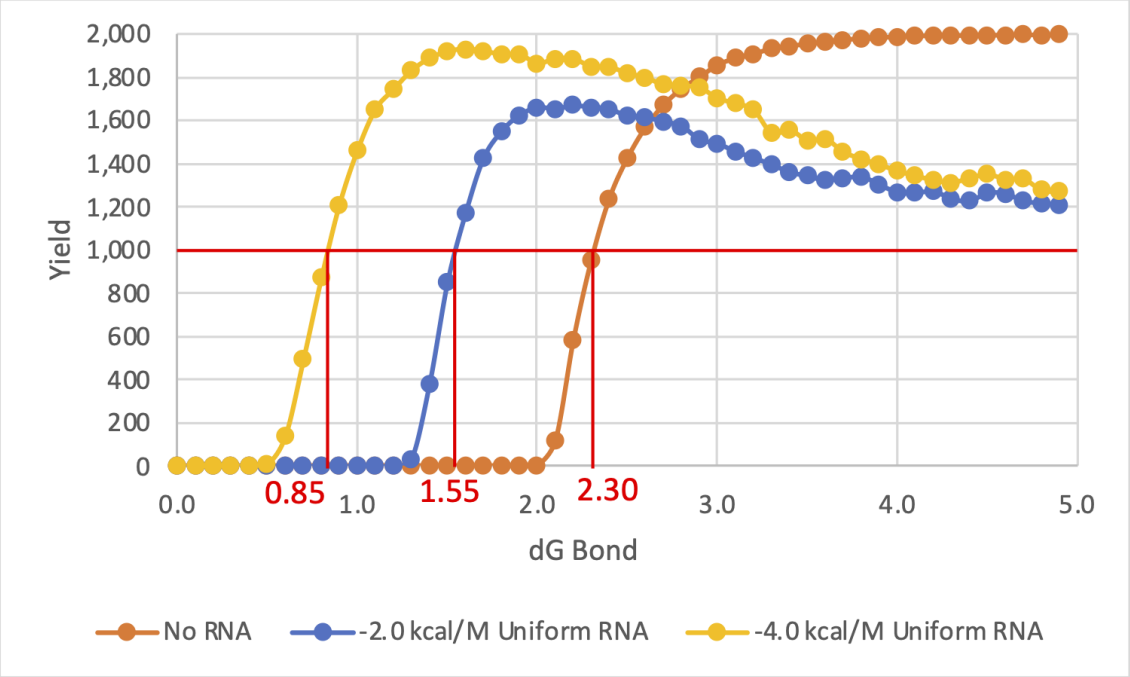

Figure 3.6: Graph showing the yield for a variety of ΔG_{prot} values for different bond strengths. Two ΔG_{RNA} bond strengths were used as well, to show the effect of changing these strengths, too. Each data point is based on the results of a single simulation that contained 2000 RNAs (when present) at temperature $298.15K$. During the simulations, 24000 capsomers were gradually ramped into a volume of $0.7\mu m^3$ at a rate of $400s^{-1}$ and the simulation ran until $t = 2000s$. The capsomer-PS binding rate was $k_f^r = 1.11 \times 10^7 M^{-1}s^{-1}$, the capsomer-capsomer 1^{st} order rate was $1.0 \times 10^6 s^{-1}$ and the capsomer-capsomer 2^{nd} order rate was $1.0 \times 10^6 M^{-1}s^{-1}$. The weaker RNA has a uniform ΔG_{rna} of -2.0 kcal M^{-1} and the stronger RNA has a uniform ΔG_{rna} of -4.0 kcal M^{-1}. The effect of increasing ΔG_{rna} can be seen, as the RNA stabilises the capsomers, allowing them to assemble in less favourable conditions.

the number of completed capsids at the end of the simulation of 2000s of chemical reactions, based on the choice of both ΔG_{prot} and a uniform ΔG_{rna}. This graph shows how adding RNA will strengthen the stability of the capsid, as predicted above. It also shows that other complications may occur: lower ΔG_{rna} does not generate as high a maximum yield within the 2000s time limit as RNA-free assembly. This could be due in part to differences in $S_U(n)$ and $S_D(n)$, which are less favourable in the presence of the RNA and play a major part in the extra term in RNA-based assembly, which suggests the RNA is generally beneficial. However, viruses need some minimal ΔG_{rna} to overcome the extra limitations caused by its presence, as is echoed by Table 3.3. A counter-argument to that would be that in this stochastic simulation, kinetic traps are available, which are not present in the equilibrium model presented here, perhaps also accounting for the lower maximal yield. Another interesting observation is that shifting the strength of the RNA-capsomer bonds in this simulation does not affect the yield as much as shifting the strength of the capsomer-capsomer bonds, which agrees with what was shown in the Tables 3.2 and 3.3 above.

3.3 SIMPLE MODEL OF QUASI-EQUILIBRIUM

Normally, in complex systems, trying to find the states that the system tends towards is important. In the case of virus assembly, all reactions are second order, which makes identifying stable states much harder, especially as most systems have a very large number of dimensions (*e.g.* a dodecahedral model with no RNA, with a 73 dimensional strongly interconnected state space and second order rates only [18]).

To simplify the calculations, we convert all of the equations to first order. The equations of motion in this case would be of the form $X(t) = X_0 e^{Bt}$ for some matrix B. This will allow identification of the stable states using eigenvalue/eigenvector methods for a range of different parameter values (including the ΔG_{prot} between capsomers).

We first have to consider the assembly of a single capsid, so exactly 12 capsomers in the system. Later, how to extend this to a larger number of capsomers will be covered.

We will continue using the assumption that if an intermediate of n capsomers (hereafter an n-mer) forms, that it can only form the most stable intermediates, as

given in Figure 3.2 and in Zlotnick's work [59]. With these assumptions in place, the system can be made first order using Partition Theory.

3.3.1 Partition Theory

Consider for the moment, a system where only 12 capsomers are present (*i.e.* enough for exactly one dodecahedral capsid) and only the lowest energy capsid intermediates are allowed to form. How many ways can the 12 capsomers arrange themselves as a collection of these intermediates?

To answer this, we look to Partition Theory, a branch of number theory, where a partition of (integer) n is a way to write n as the sum of a number of positive integers. Only the quantities, not the order, of these positive integers matter. The number of partitions for n is written as $p(n)$. As an example, the partitions of 4 are:

$$\begin{aligned} &4 \\ &3+1 \\ &2+2 \\ &2+1+1 \\ &1+1+1+1, \end{aligned}$$

which means that $p(4) = 5$, as there are 5 different ways to write 4 as the sum of positive integers.

3.3.2 Partitions Of 12 Capsomers

To apply partition theory to our dodecahedral models, we look at the assembly of a single capsid. We allow these capsomers to form up to 12 intermediates (including free capsomers as intermediates), with each n-mer being the most stable structure for that size. As capsomers can only be added as whole units and all n-mers are equivalent to other n-mers of the same size, this is effectively a partition of 12 capsomers. As $p(12) = 77$, we have a system with 77 states. The system is allowed to move between states only via the addition/removal of single capsomers, creating a graph of 77 states that the system can traverse through. This graph can be seen in Figure 3.7.

The transition rates between states in this system can be calculated, as each state will have a fixed set of transitions and each transition has an associated reaction

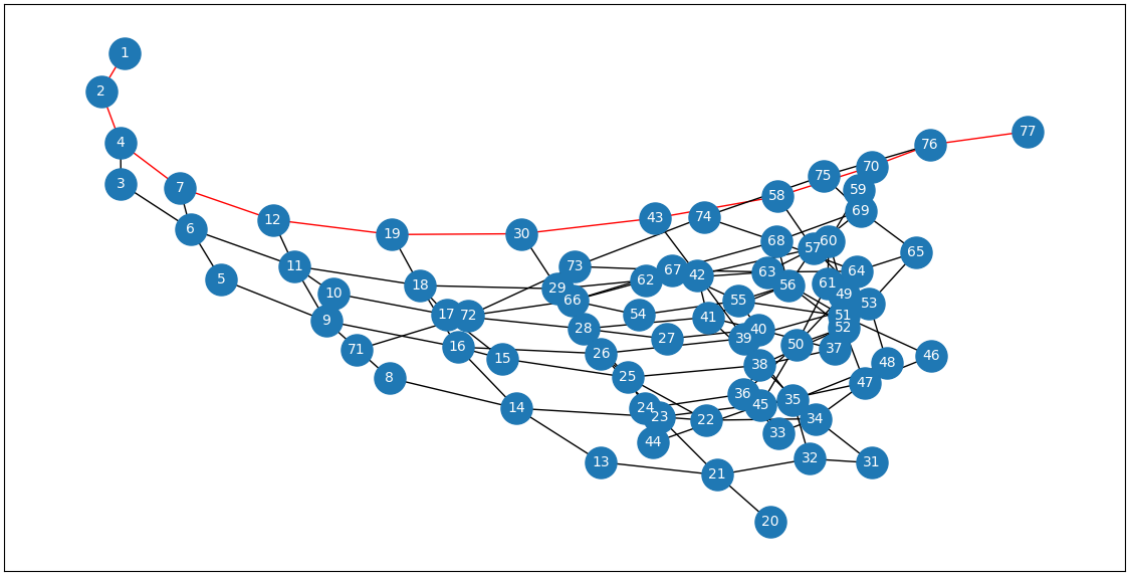

Figure 3.7: This graph shows the partitions of 12 capsomers (with some shown in Table 3.4), with edges added when a single bond formed/broken would result in a transition between partitions. For the case where only one intermediate other than free capsomers was present, the edges were highlighted in red.

rate for it. Through this, we can define a vector $\mathbf{x}$, where x_i contains the probability of the system being in state i, with $\sum_{i=1}^{77} x_i = 1$. With this we define:

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x}, \tag{3.53}$$

where A is a constant matrix containing the transition rates. $A_{ij}, i \neq j$, represent the rate of the reaction for the transition from state j to state i. We also have $A_{ii} = -\sum_{j=1,j\neq i}^{77} A_{ji}$, *i.e.* the state's probability should decrease. The rates depend on structural factors from Table 3.1 and what reaction is occurring. If it is a nucleation reaction where two capsomers form a dimer, then $(1)+(1) \rightarrow (2)$ gives us a rate of $A_{ij} = \frac{[1]\cdot([1]-1)}{2} k_f$, where $[n]$ is the number of an n-mer in the system. For an elongation equation, where an n-mer bonds to a free capsomer, then $(n)+(1) \rightarrow (n+1)$ gives us a rate of $A_{ij} = [1][n]k_f \frac{C_{fac}}{V}$, where $C_{fac} = \frac{10^{21}}{N_A} \approx 1.66 \times 10^{-3} \mu m^3 M^{-1}$ is a conversion factor, N_A is Avogadro's Number and V is the volume. This conversion factor is necessary as we consider the number of units per state, rather than concentration. It relates the number of molecules per unit volume to moles per unit volume (also known as molarity). For a removal equation, where a capsomer leaves an n-mer, then $(n) \rightarrow (n-1)+(1)$ gives us a rate of $A_{ij} = [n]\frac{S_D^P(n)}{S_U^P(n)} e^{c(n)} k_f$.

Equation 3.53 shows us that despite many of the individual reactions at the protein level being second order, we have a first order system. Using the matrix A, we can find the eigenvalues, λ_i and the eigenvectors $\hat{\mathbf{x}}_i$, so

$$A\hat{\mathbf{x}}_i = \lambda_i \hat{\mathbf{x}}_i. \tag{3.54}$$

If we have $\lambda_j = 0$, then we get

$$\frac{d\hat{\mathbf{x}}_j}{dt} = A\hat{\mathbf{x}}_j = \lambda_j \hat{\mathbf{x}}_j = 0 \tag{3.55}$$

and so $\hat{\mathbf{x}}_j$ is a steady state for the system. This would indicate the probability that each state would form in the system.

For the system $\frac{d\mathbf{x}}{dt} = A\mathbf{x}$, we can write a general solution:

$$\mathbf{x}(t) = \sum_i C_i e^{\lambda_i t} \hat{\mathbf{x}}_i, \tag{3.56}$$

with some constant C_i and the eigenvalues and eigenvectors as before. If we have $\lambda_i \leq 0$ for all i, then this means all non-zero exponential terms will decay and the system will tend towards one of its steady states.

Theorem 3.3.2.1. *Matrix A (as given in Equation 3.53) must have at least one eigenvalue equal to zero.*

Proof. Preserving mass action requires all of the diagonal elements to satisfy $A_{ii} = -\sum_{j=1,j\neq i}^{77} A_{ji}$. This can be rearranged to give $\sum_{j=1}^{77} A_{ji} = 0$ for each i. Thus, the columns are not linearly independent and $\det A = 0$. As the determinant can be written as the product of eigenvalues: $\det A = \prod_{i=1}^{77} \lambda_i$, at least one $\lambda_i = 0$. □

Using code produced during this PhD, along with the LAPACK FORTRAN package [2], A is generated, the graph in Figure 3.7 is created and the eigenvalues and eigenvectors are found. As LAPACK is a numerical solver, it returns all results as a floats, which may be subject to round-off errors. As a result, any eigenvalue with an absolute value less than 10^{-12} is treated as zero. From Theorem 3.3.2.1, we know that there must be at least one eigenvalue equal to zero, so if there are no sufficiently small eigenvalues, the smallest eigenvalue is treated as a zero with a larger than normal round-off error. A very small number of eigenvalues were found to be > 0. However, this was always the eigenvalue with the smallest absolute value. Also, they all had an absolute value of $\approx 10^{-12}$ or less, so they are likely to be zeroes

with a small positive value due to roundoff errors. All other eigenvalues satisfied $\lambda_i \leq 0$. As our system satisfies $\lambda_i \leq 0$ for all i, it will eventually reach a steady state (*i.e.* the eigenvector that corresponds to a zero eigenvalue) that describes the probability that the system occupies any individual partition.

It is intuitive that this should be the case but no proof will be offered here. As $\mathbf{x}$ must satisfy $\sum_{i=1}^{77} x_i = 1$ due to $\mathbf{x}$ being a vector of probabilities and $\mathbf{x}(t) = \sum_{i=1}^{77} C_i \hat{\mathbf{x}}_i e^{\lambda_i t}$, we find that if we have a $\lambda_j > 0$, then this eigenvalue's exponential term will increase out of control as t increase, unless either $C_j = 0$ or $\hat{\mathbf{x}}_j = \mathbf{0}$. Otherwise this would lead to $\mathbf{x}$ growing large and no longer satisfying the above condition.

The code generated the eigenvalues for this system for a range of ΔG_{prot} values, from 2.0 to 6.0 with a step size of 0.2. Following this, additional eigenvalues were generated for $\Delta G_{prot} = 3.9$ and $\Delta G_{prot} = 4.1$, as this was a region of interest. All eigenvalues with an absolute value less than 10^{-3} were collected and they were sorted based on their eigenvectors to give Figure 3.8, giving a small number of eigenvalues of interest and four distinct section on the graph.

Of the eigenvectors of interest in the system, two (QSS1 and QSS2) do not represent probabilities but instead represent flux between two states, as they both have terms where $\hat{\mathbf{x}}_i > 0$ and also terms where $\hat{\mathbf{x}}_i < 0$ for non-negligible magnitudes of $\hat{\mathbf{x}}_i$. If all values of $\hat{\mathbf{x}}_i$ were positive, this could be interpreted as a probability vector and if all values of $\hat{\mathbf{x}}_i$ were negative, they could be scaled by -1 to give a probability vector, as eigenvectors are invariant to scaling. However, due to them having opposite sign, they cannot be scaled to represent probabilities. Only the zero eigenvalue states satisfy either $\hat{\mathbf{x}}_i \geq 0$ or $\hat{\mathbf{x}}_i \leq 0$, which leaves three steady states that the system can exist in, depending on the value of ΔG.

To ensure that the model is giving useful results, we need to know that the system will not allow the probabilities to become negative. To start, we consider A_{ij} and note that the only negative terms present are on the diagonal, *i.e.* A_{ii}. So within $\frac{dx_i}{dt} = \sum_{j=1}^{77} A_{ji} x_j$, the only term with a negative coefficient is $A_{ii} x_i = -|A_{ii}| x_i$. Thus, on the boundary, where $x_i = 0$, there can be no negative term in $\frac{dx_i}{dt}$ and therefore the value of x_i cannot cross $x_i = 0$. So, if $x_i(t_0) \geq 0$ for all i, then $x_i(t) \geq 0$ for all $t > t_0$.

The graph in Figure 3.8 has four distinct sections, Section A is where ΔG is sufficiently small that the most probable state is just free capsomers. In Section C, ΔG_{prot} has increased sufficiently that the most probable state is a completed capsid.

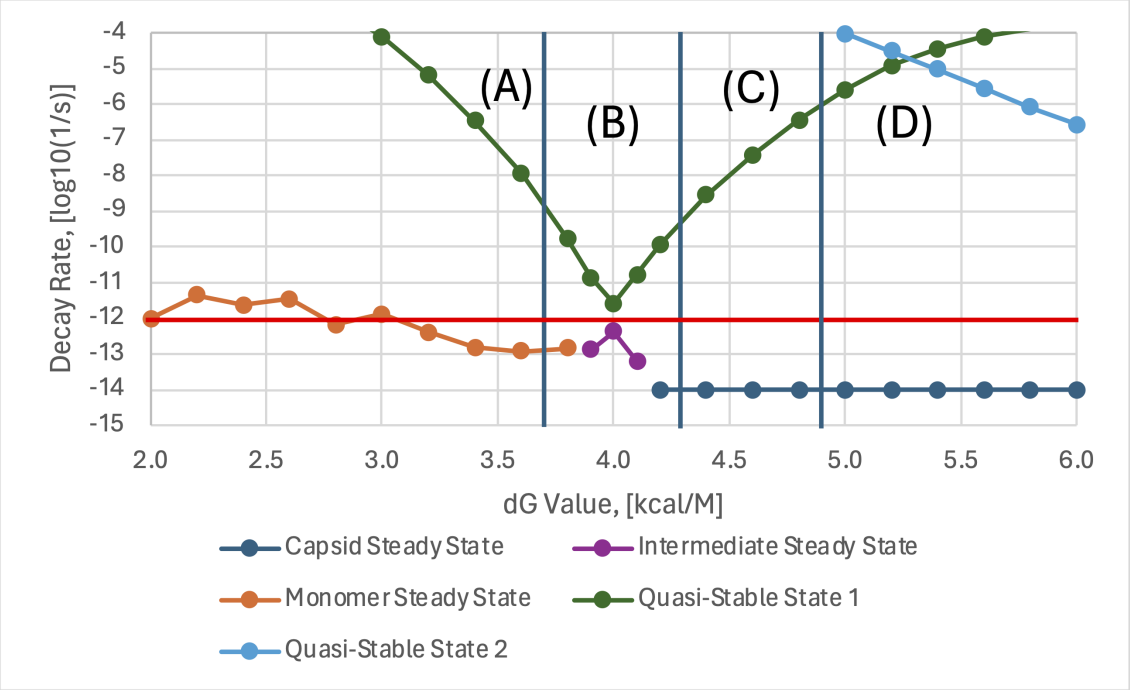

Figure 3.8: A graph indicating the relative values (given as log_{10}(abs(eigenvalue))) of the smallest eigenvalues present in the system, colour coded to indicate their respective eigenvectors and featuring four sections. The four sections can be summarised as follows: (A) Unassembled Capsid, (B) Transitory Section, (C) Stable Capsid and (D) Kinetic Trapping. For a small number of eigenvalues, LAPACK returned precisely zero, so to ensure that these could be placed on the graph, they were treated as being 10^{-14}. The eigenvalues for 5 significant eigenvectors are shown in the graph. The Capsid Steady State represents the eigenvector with a 100% probability to occupy the state where the capsid is complete. The Monomer Steady State represents a 100% probability that the capsomers are all disassembled. The Intermediate Steady State is interesting, as its eigenvector transitions from the eigenvector for the CSS to the eigenvector for the MSS, with the probability flowing between the two states as ΔG_{prot} increases. Relevant elements of the eigenvalues in this state are shown in Figure 3.9. The two Quasi-Steady States' eigenvectors represent flux between two states, as they contains both positive and negative components and have larger eigenvalues than the steady states. QSS1's eigenvector represents the flow of probability needed to transition between the CSS and MSS eigenvectors and it is at its lowest point (and so its slowest decay time) midway through the transition between the two. QSS2's eigenvector represents a flow between completed capsids and two separate half capsids. It appearing when it does, suggests that with a greater ΔG value, more states may be able to have a stable existence, especially as, for $\Delta G = 6.0$, two other eigenvalues (not shown) of size $< 10^{-4}$ formed, suggesting that other quasi-stable states similar to QSS2 may appear for larger parameter values.

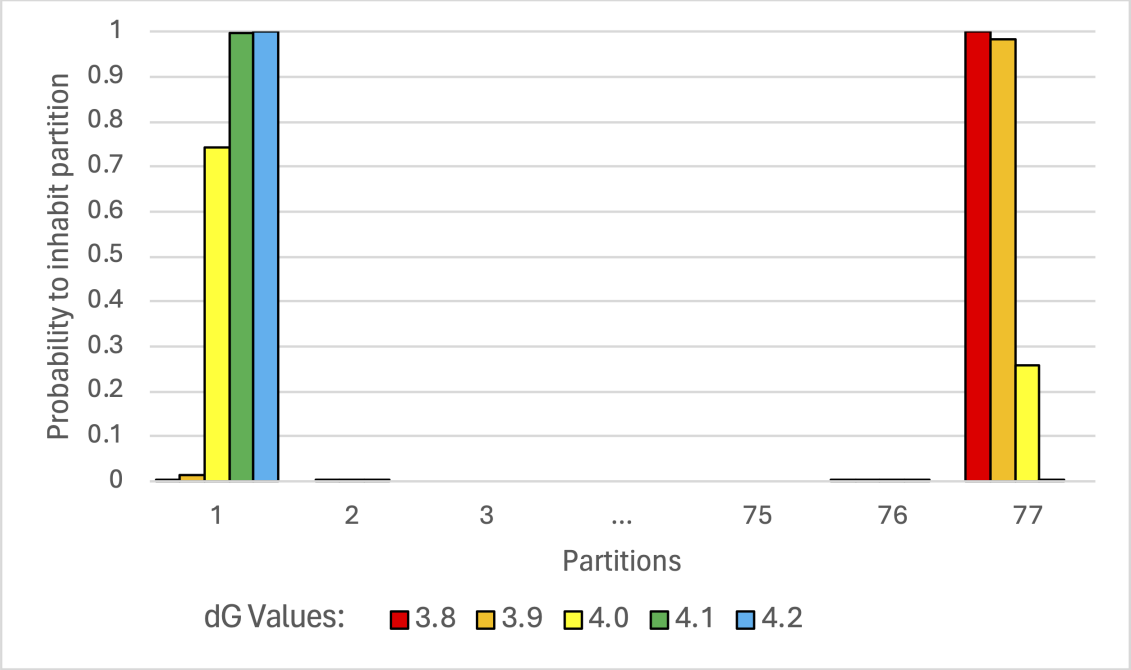

Figure 3.9: Bar chart to show key regions of eigenvectors in the transitory section of Figure 3.8. It shows the i^{th} element of the current steady state eigenvector, $\hat{x}_S^i$ for $i \in \{1, 2, 3, 75, 76, 77\}$, giving the probability that the system will occupy partition i, as shown in Table 3.4. All values omitted are less than $\approx 10^{-4}$. As ΔG_{prot} increases, probability flows from the Monomer Steady State (where $\hat{x}_S^{77} = 1$) into the Capsid Steady State (where $\hat{x}_S^1 = 1$). The bars for $\Delta G = 3.8\text{kcal}M^{-1}$ and $\Delta G = 4.2\text{kcal}M^{-1}$ represent the Monomer and Capsid Stable States, respectively.

Next, we consider Section B, which is a transitionary region. In it, eigenvectors for the monomer steady state transition to become the capsid steady state via the intermediate steady state, which has a non-zero probability of either partition representing the system, as shown in Figure 3.9. Additionally, Section D is the region where kinetic trapping becomes more probable and a second QSS appears, which may lead to another change to the stable state for sufficiently large ΔG_{prot}. Notably, two more states appear in the graph's range at $\Delta G_{prot} = 6.0$, which could suggest even more kinetic trapping to come. Based off the graph, the critical value of ΔG_{prot} can be estimated as $\Delta G_c = 4.0$.

This means that for low ΔG values, the system's most stable state is disassembled. However, for larger ΔG values, the most stable state is fully assembled and there is a period of transition between them. This makes sense, as a virus needs to be on the knife's edge between assembly and disassembly, so that when it reaches a new host

cell it can release its genetic information and begin reproduction. QSS1 represents the flux of probability from one state in the system to another. It makes sense that it is long lived for values of ΔG_{prot} where the system's probabilities transition, so a virus would likely evolve so that it would avoid ΔG_{prot} values in this transitionary region, to ensure it reaches a steady state faster. For higher ΔG_{prot} values, QSS2 becomes longer lived. However, this is still a slow effect as there is not a second stable state forming. It would make sense for this to be the limit of the validity of this model, as in a cell, there would be more capsomers to make up a capsid than are present in this model.

This system could be extended to a system with a larger number of capsomers by choosing a number N (choosing N as a multiple of 12 is sensible, as this means there will be a state with $\frac{N}{12}$ completed capsids) of capsomers to be present, finding all partitions of N and excluding all partitions with integers > 12. Using these partitions, another graph of states could be formed (like in Figure 3.7) and a new matrix A of the rates could be generated. Then, using the same process as for $N = 12$, stable states for the larger system can be found.

Table 3.4: Table indicating a selection of the 77 partitions of 12. When used for viruses, each column represents the number of n-mers (intermediates with n capsomers) present in that state. Important partitions are 1 and 77, which represent the complete capsid and fully disassembled capsomers, respectively.

Partition	1	2	3	4	5	6	7	8	9	10	11	12
1												1
2	1										1	
3		1								1		
4	2									1		
7	3								1			
12	4							1				
19	5						1					
20						2						
21	1				1	1						
30	6					1						
43	7				1							
44				3								
45	1		1	2								
58	8			1								
59			4									
60	1	1	3									
70	9		1									
71		6										
72	2	5										
75	8	2										
76	10	1										
77	12											

— 4 —

Generalised Stochastics Models For Simulating Capsid Assembly

In this chapter, initially the issues with further extending the ODE methods from the previous chapter are presented. The advantages of using Gillespie methods instead are then presented. The rest of the chapter discusses a general method that was developed to create a description of a given capsid's structure using only a small selection of its surface. Then, the way that this general method is integrated with existing methods for modelling virus assembly models is discussed, followed with ways to improve those methods in this case.

For larger viruses that contain many capsomers, techniques other than ODEs and equilibrium models must be used to model the assembly of viruses. The reason for this is covered in more detail in Section 4.1.

This means we need another way to model the assembly. Many issues with ODE models can be solved by using a stochastic simulation, such as a Gillespie algorithm. Gillespie algorithms, as covered in Section 2.2, allow the reaction rates for the system to be calculated on the fly. To do this, the algorithm should know the current state of each capsid and be able to determine the potential reactions that may occur. For this purpose, two graphs are needed. One that indicates where capsomers sit relative to other capsomers, indicating capsid geometry and another graph that determines where the RNA can sit relative to the capsomers, indicating potential RNA geometry. However, these two graphs are quite time consuming to generate by hand, so an algorithm to rapidly generate these graphs from a minimal amount of user friendly input was developed. Section 4.2 covers how this algorithm works and also how much information is needed.

In order for the program to identify all potential reactions and to choose one to fire, we need another algorithm. The potential reactions are identified, then a Gillespie algorithm is used to determine which will fire. It implements Binary Trees to increase the speed with which it selects which intermediate will react, to minimise the number of calculations necessary. Then it chooses which of this capsid's reactions will occur using SSA. This process is covered in Section 4.3.

Collectively, this chapter gives a stochastic simulation model of the assembly for a range of viral models.

4.1 ISSUES WITH USING ODE METHODS

For large viruses, which contain many capsomers, ODE and equilibrium models are less useful than they were in Chapter 3. To form a set of ODEs, knowledge of all intermediates and the reactions describing their assembly is needed. For a capsid assembled from 12 pentameric capsomers (with no RNA), there exist 73 different intermediate structures and their interaction network is shown in Figure 4.1. When RNA (with 12 PSs that all bind to a capsomer) is introduced to this system, there are a total of 85,376 different intermediate structures [18]. With this many states, is should be obvious that forming a set of ODEs to model a system where RNA is present would be unfeasible. If viruses made of more than twelve capsomers are considered, the number of states is even larger. For example, within some proposed assembly pathways STNV (covered in Chapter 6) has over 10^{12} potential RNA arrangements within a completed capsid. This number does not consider the RNA arrangements within assembly intermediates. Alternatively, a small subsection of these states could be chosen and modelled (as Zlotnick did for the dodecahedral model without RNA [59], covered in more detail in Chapter 3). Unfortunately, this would ignore a significant number of states that could contribute to assembly and it also requires the assembly pathway to have been chosen in advance. Simulations of this chosen pathway would not identify kinetic traps that the assembly might encounter either.

One workaround to this is to use a Gillespie algorithm. This allows the transitions to be calculated as the simulation goes on, so a smaller number of states need to be considered. As Gillespie samples the most probable states, the states considered are likely to be relevant to the assembly process.

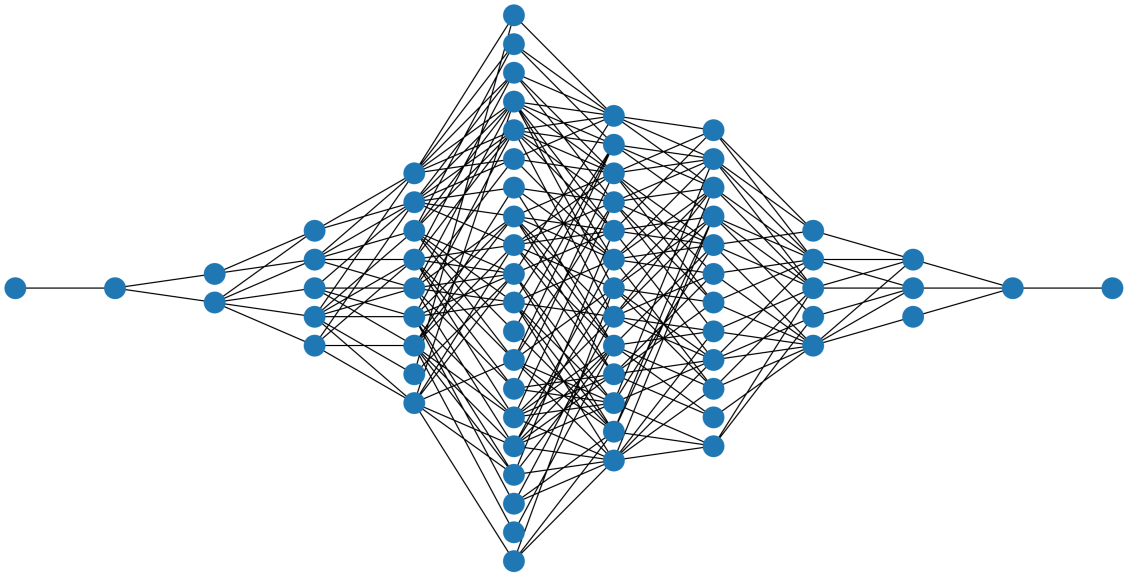

Figure 4.1: A network that shows all intermediates for a dodecahedral capsid as nodes and the reactions that can transition between them as edges. The leftmost node is a free capsomer and the rightmost node represents a fully assembled capsid. All nodes in one column have the same number of capsomers present and each reaction only adds/removes one capsomer.

4.2 GENERALISED TILING MODEL

To identify which reactions may occur, the structure of the intermediate must be known. This means it is necessary to have a graph that contains all of the connections between capsomers and the free energy (ΔG_{bond}) interactions between them, which will be called the capsomer graph. This capsomer graph is a form of Adjacency Matrix, though if it was stored as a matrix, this would be a sparse matrix. It is also sometimes called the "interaction network" [21] and it describes which capsomers

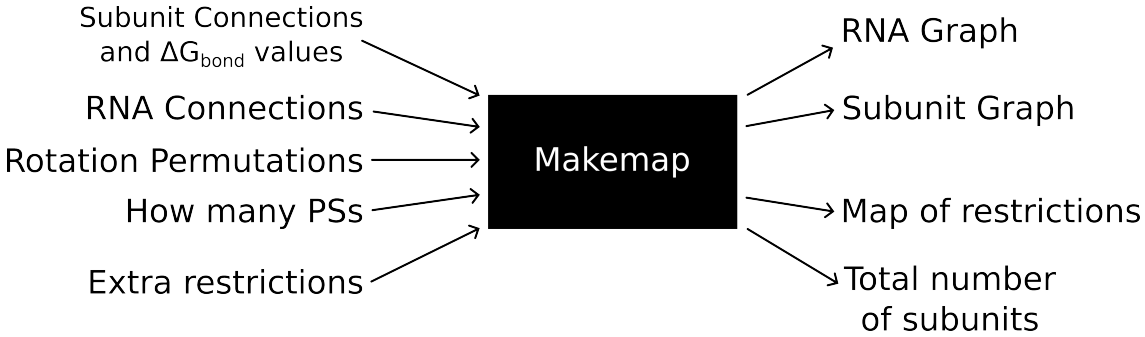

Figure 4.2: A flow diagram to summarise the code described in Section 4.2.

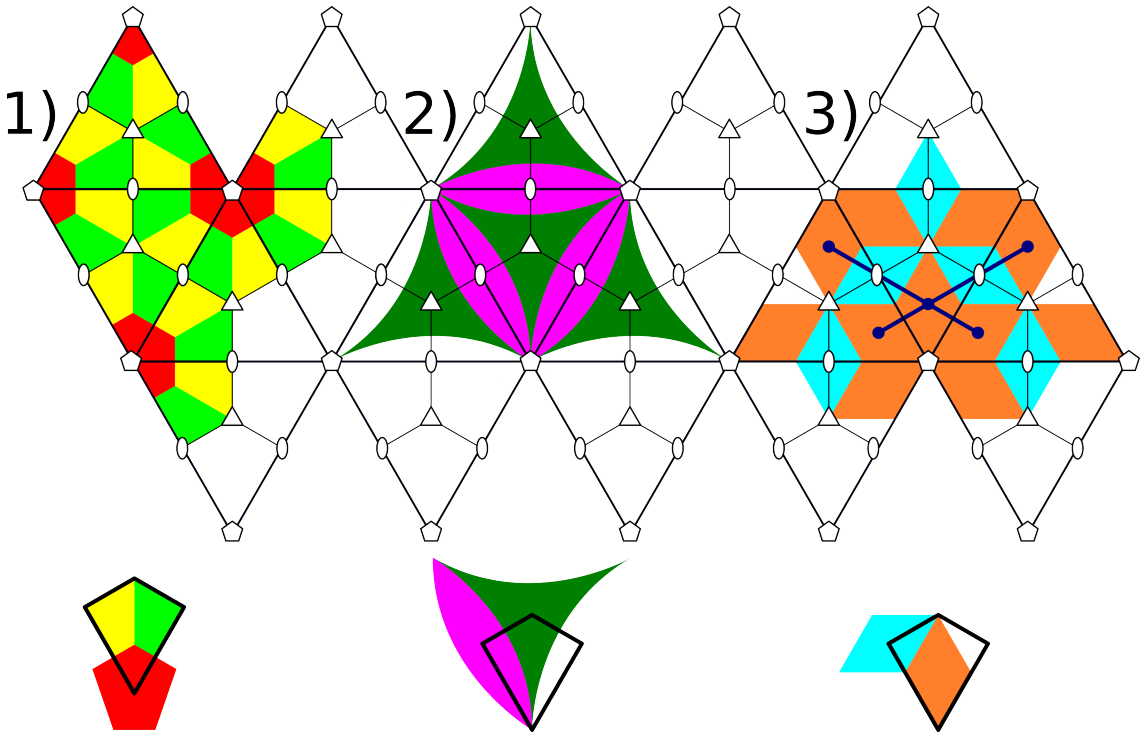

Figure 4.3: Three examples of potential tilings of the capsid, shown as patches of tilings, to show how little information is needed for tile files. The 5-,3- and 2-fold axes are represented by black and white pentagons, triangles and ovals, respectively. Below each patch is an asymmetric unit, with one copy of each distinct capsomer shown, including ones that sit upon symmetry axes. The algorithm would only need the information for these capsomers and could then generalise it to the entire capsid, using icosahedral symmetry. In Example 1, there are multiple capsomers present in one Fundamental Domain (FD). In Example 2, all capsomers sit on symmetry axes (specifically the 2-fold and 3-fold axes). Example 3 illustrates how an average Rhomb $T = 3$ virus would be tiled, including RNA that can go around the 5-fold axis and across both nearby 2-fold axes. The navy lines indicate the four edges on one vertex of a potential RNA graph. These lines could be extended to cover the whole capsid and thus form the RNA graph for the capsid. In Appendix C, it is detailed how these become tile files for the algorithm to understand. For simplicity, consider Examples 1 and 2 without RNA.

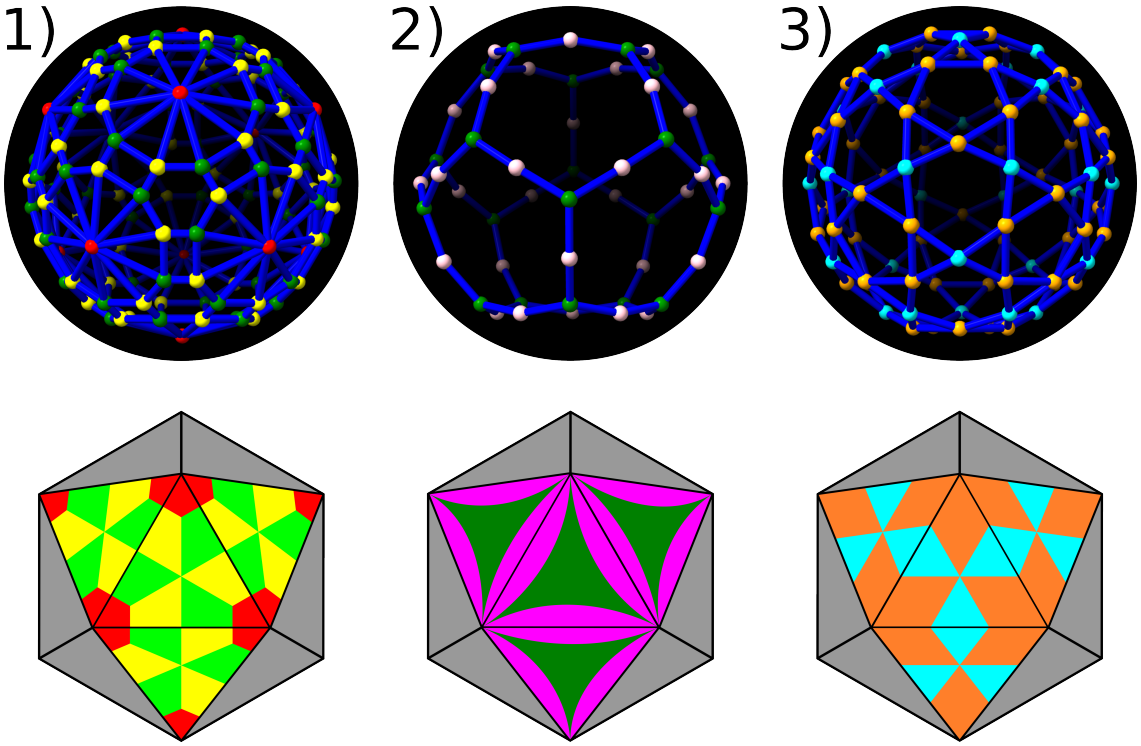

Figure 4.4: A 3D rendering (produced via the process described in Section 5.3) of the capsomer graphs produced using tile files for capsids illustrated in Figure 4.3, using approximately the same colour scheme for each capsid. Part of the tilings from that same figure have been applied to an icosahedron below the 3D rendering as well. Each node represents one capsomer and each blue edge represents a capsomer-capsomer interaction between the two nodes. The RNA graph for Example 3 is not shown here, for simplicity. This should help visualise how these graphs look and the complexity of them that leads to this algorithm being necessary.

sit close together. It is used instead of tilings in protein nanotechnology, where some protein cage structures cannot be modelled as tilings. The interaction network abstracts from the exact shape of the capsomer and only retains information on inter-capsomer interactions.

In this thesis' capsomer graphs, each node's edges are stored as a list of adjacent nodes. When RNA is present, we will also need a graph that can be used to map out the positions between two consecutive PSs on the RNA. Thus, the RNA graph represents the many paths that the RNA can trace out on the interior of a completed capsid. An example of what this looks like is shown in Figure 4.5. This is also stored as a list of adjacent nodes. Both of these graphs are complicated and time consuming to produce by hand, so a way to generate these computationally is useful.

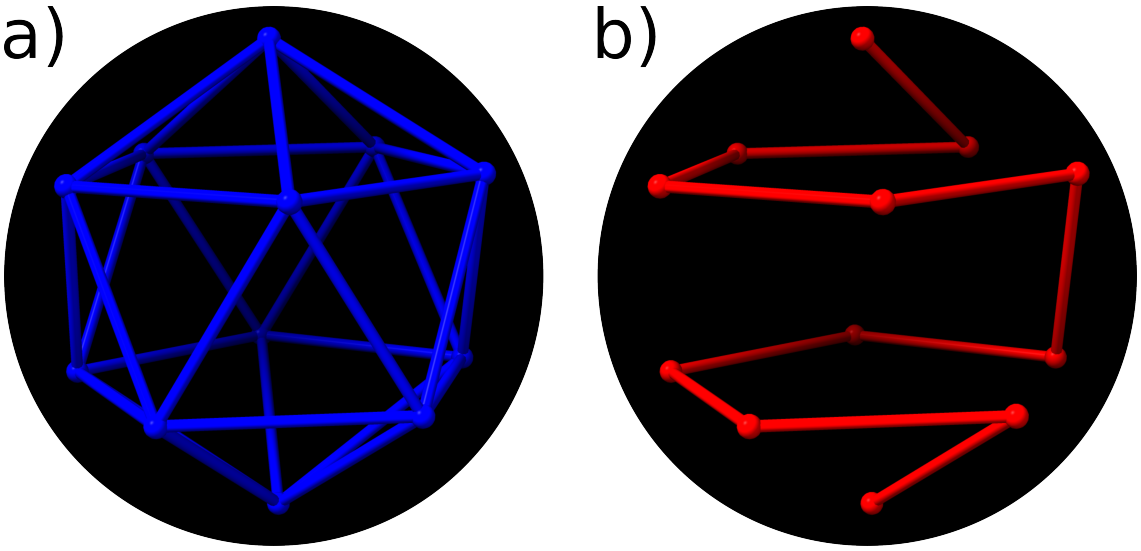

Figure 4.5: (a) The RNA graph for a dodecahedral capsid made of 12 pentamers. Vertices indicate binding sites for the RNA and edges indicate where a PS, adjacent (along the RNA) to the PS sat at that vertex, can sit. (b) An example of a Hamiltonian Path on this RNA graph.

Consider Figure 4.3, which is a 2D net of an icosahedron with three example tilings of potential viruses on it. Each colour represents one distinct capsomer in different positions on the capsid. These capsomers may be identical to each other, they may be made of the same proteins in a different configuration or they may be made of completely different proteins. Being able to simulate the assembly of a wide range of proteins such as these is important, to develop an understanding of viral assembly. These simulations take place in the presence or absence of RNA. So a method to generalise the input of various assembly dynamics for simulations is useful.

A program was created that takes an input file with the information for one Fundamental Domain (FD) of a capsid and will generate a complete map of the capsid by taking advantage of the icosahedral symmetry of the capsid. Firstly, consider the symmetry of the icosahedron.

An icosahedron is a platonic solid made up of twenty identical equilateral triangles and it has 60 rotational symmetries. Viruses use this symmetry to minimise the number of distinct capsomers that they need to encode, whilst maximising the size of the capsid, so that it can fit enough genetic information to encode those proteins plus other components required by the virus (known as the principle of genetic economy).

Viruses tend to utilise icosahedral symmetry, rather than tetrahedral or octahedral symmetry. The corresponding alternative shapes give a smaller internal volume for a protein of a given size, so icosahedrons are preferred. As a result of the icosahedral symmetry, the program can take one capsomer as an input and then generate a graph indicating the relative positions of all other copies of that capsomer.

On a net of an icosahedron, the fundamental domain can take a wide range of shapes. Within this thesis, we will draw a Fundamental Domain (FD), which we choose to be kite shaped, sitting with one of its vertices at a 5-fold axis, one at a 3-fold axis and two at 2-fold axes, as is usual in the literature. They cover $\frac{1}{60}^{th}$ of the entire capsid's surface and contains all the information needed to describe the capsid. An illustration of a fundamental domain tiling the net of an icosahedron is shown in Figure 4.6.

For this algorithm to operate, it needs an input of every distinct capsomer and its neighbours. Here, two capsomers are called distinct if they are not rotationally equivalent within icosahedral symmetry. Different tile shapes correspond to different types of biological entities. In addition, identical tile shapes in symmetry-inequivalent positions can also be realised by different structures. However, typically this would be different conformers of the same capsomer, rather than different capsomers. Example 3 in Figure 4.3 has two distinct capsomers, with their positions shown in orange and blue. These two capsomer positions both contain the same capsomer but are not rotationally equivalent to each other. However, as each orange capsomer position is rotationally equivalent to all of the others, these only represent one distinct capsomer. For each capsomer where there are 60 copies of it in the completed capsid, the algorithm will consider it as sitting within a fundamental domain. The algorithm considers any capsomer that sits on a symmetry axis as not being within a fundamental domain, as it will have a reduced number of copies and will exhibit a reduced number of symmetries, so will be treated slightly differently by the algorithm.

In Figure 4.6, we show the numbering scheme used to identify each fundamental domain and each symmetry axis in the algorithm. Within the algorithm, fundamental domains take the numbers 1-60, 2-fold axes take the numbers 61-90, 3-fold axes take numbers 91-110 and 5-fold axes take 111-122. This leaves every region in the capsid with a unique fundamental domain/axis number.

For the algorithm to produce the two graphs, it should have an input that will give all of the information needed whilst also being understandable for a person to read and generate. Details of the file format can be found in Appendix C.

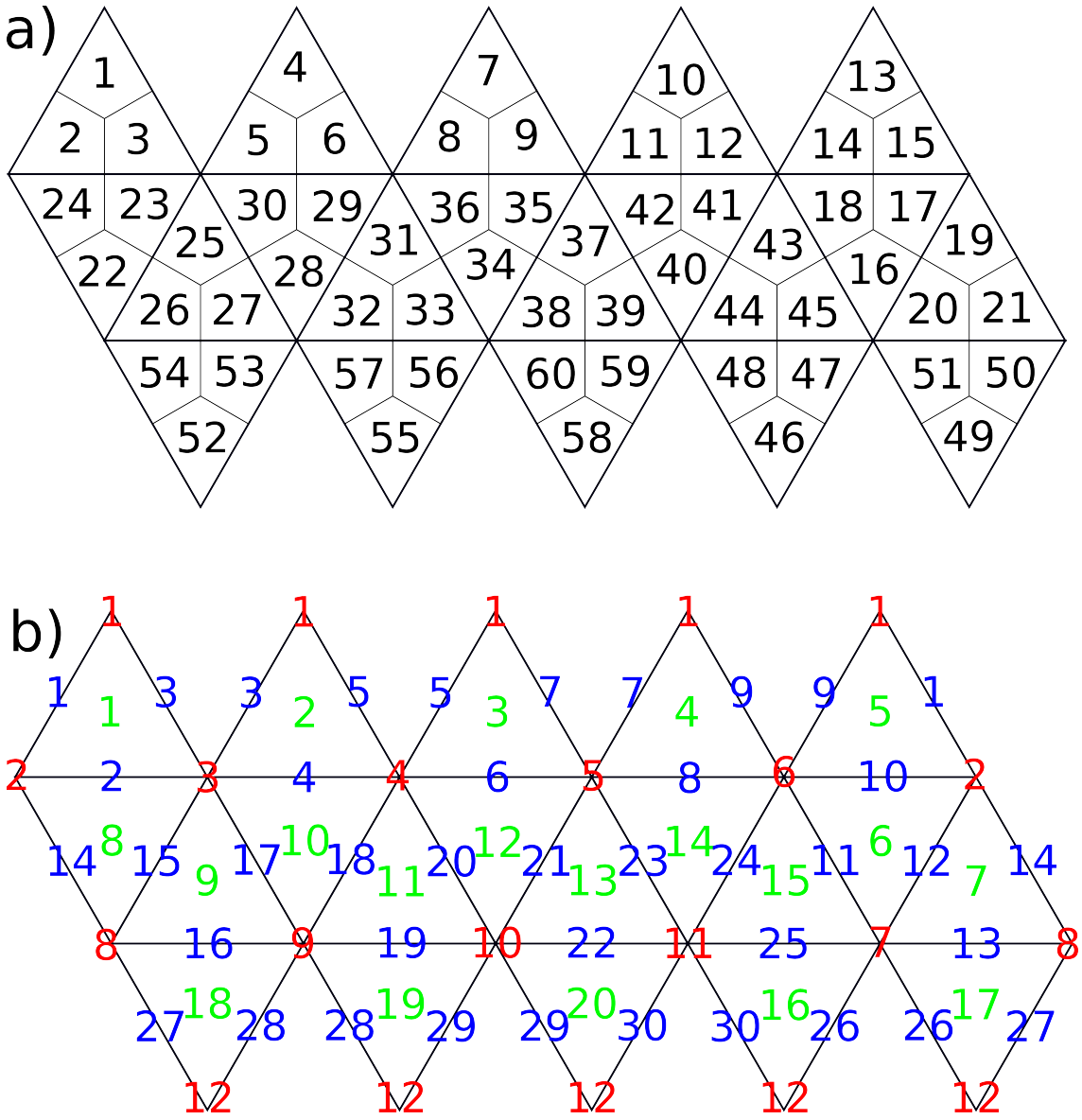

Figure 4.6: Planar nets of the icosahedron, split up so that each fundamental domain and axis can be numbered. There are 2-fold axes at the midpoint of edges, 3-fold axes at the centres of the triangles and 5-fold axes at each vertex. Fundamental domains have been chosen as kite shapes for the purpose of this numbering. (a) Shows the Fundamental Domains and assigns a number to each of them. (b) Numbers the 5-fold (red), 3-fold (green) and 2-fold (blue) symmetry axes. When multiple positions have the same number in the same colour, this just indicates that when the net is folded, they are in the same position. The two figures line up, so fundamental domain 1 would sit between 5-fold 1, 3-fold 1 and 2-folds 1 and 3.

The following information is needed so that the algorithm knows everything about each FD. It needs to know how many distinct capsomers are present, what they are made of, how many interactions each of these capsomers make, what these interactions' free energy is and where the interactions occur. For simulations with RNA, it needs to know what interactions the capsomers can have with the RNA, what paths the RNA can trace out, what extra restrictions on the RNA (*e.g.* mutual exclusion, as implemented in Chapter 6) are present and even how many PSs does the RNA have. The tile file contains this information in an easy to read and produce format.

Once a tile file has been made, the algorithm needs to interpret this information into a form it can use for simulations. Instead of using two numbers (FD, the number given in Figure 4.6 and PN, the protein identification number of a capsomer in an FD) to refer to each capsomer's position in the capsid, it would be easier to give a single number that refers to the capsid positions, so arrays of information can easily be stored. This numbering must be unique for each capsomer position, should not have any gaps and should scale to any capsid the program could be given. For the first step of doing this, we need to define a variable:

$$protNoMax(i) = \begin{cases} \text{if } i = 1: & \#\text{ of distinct proteins that sit} \\ & \text{in a fundamental domain} \\ \text{if } i = 2: & \#\text{ of distinct proteins that sit} \\ & \text{on a 2-fold axis} \\ \text{if } i = 3: & \#\text{ of distinct proteins that sit} \\ & \text{on a 3-fold axis} \\ \text{if } i = 4: & \#\text{ of distinct proteins that sit} \\ & \text{on a 5-fold axis} \end{cases} . \tag{4.1}$$

This gives the number of distinct capsomers that sit in either a FD or on each of the three axis types. With this, we can define a map that takes the fundamental domain (which takes a value in the range $1 \leq FD \leq 122$) that a capsomer sits in and the protein capsomer number ($PN \in \mathbb{N}_{\neq 0}$) for that capsomer and gives a unique identifier ($UI \in \mathbb{N}_{\neq 0}$) for that capsomer position.

$$G : [1 : 122] \times \mathbb{N}_{\neq 0} \to \mathbb{N}_{\neq 0}. \tag{4.2}$$

We call this map, getNum(FD,PN), which only relies on its input and the values of protNoMax, which are fixed for a given capsid. This function is defined in Algorithm 1. Using the input FD, it identifies if it sits in a FD or which axis the capsomer sits on. This map numbers each of the protNoMax(1) protein capsomers that sit within a fundamental domain with numbers from 1 to protNoMax(1) $*$ 60, each of the protNoMax(2) capsomers that sit on a 2-fold axis with numbers from 1 + protNoMax(1) $*$ 60 to protNoMax(1) $*$ 60 + protNoMax(2) $*$ 30 and so on. This means that it will always start at 1 and not allow any overlaps in the numbering.

Examples of this numbering can come from Figure 4.3. First, consider Example 1, where there are two capsomers in each FD and one on each 5-fold axis. Thus, the green capsomers would take the odd numbers from 1-120 and the yellow capsomers would take the even numbers from 1-120. The red capsomers would take numbers from 121 to 132. Next, in Example 2 there are two distinct capsomers, the green one sitting on 3-fold axes and the pink one sitting on 2-fold axes. The pink capsomers would be numbered from 1 to 30 and the green capsomers from 31 to 50. Lastly, in Example 3, there are orange capsomers which each sit within a FD, with only one capsomer per FD and so will take a number from 1-60. It also has blue capsomers that each sit alone on 2-fold axes and take a number from 61-90.

Algorithm 1 Function to give each capsomer position on a capsid a unique number, based on the number of capsomers in each fundamental domain.

FD, an integer in [1, 122]
PN, an integer where $PN > 0$
procedure GETNUM(FD, PN)
  if $0 < FD \leq 60$ **then**
    return ((FD − 1) $*$ protNoMax(1) + PN)
  else if $60 < FD \leq 90$ **then**
    return (60 $*$ protNoMax(1) + (FD − 61) $*$ protNoMax(2) + PN)
  else if $90 < FD \leq 110$ **then**
    return (60$*$protNoMax(1)+30$*$protNoMax(2)+(FD−91)$*$protNoMax(3)+PN)
  else if $110 < FD \leq 122$ **then**
    return (60 $*$ protNoMax(1) + 30 $*$ protNoMax(2) + 90 $*$ protNoMax(3) + (FD − 111) $*$ protNoMax(4) + PN)
  end if
end procedure

Additionally, we need another function permutations($permNo, FD$), which re-

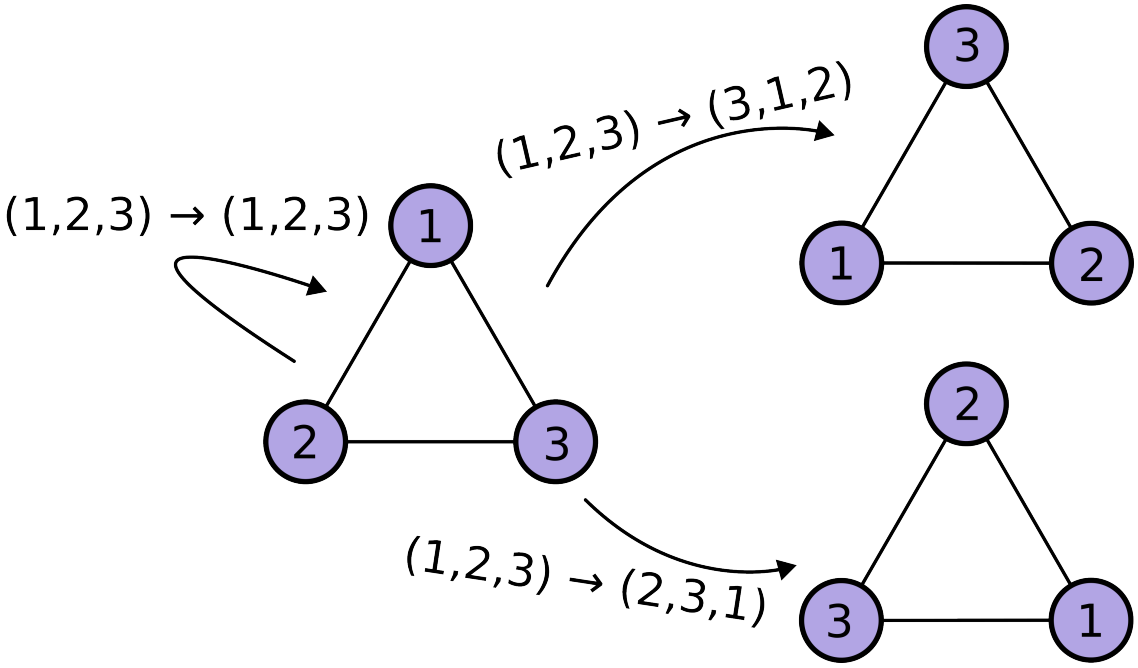

Figure 4.7: A graph with triangular symmetry. There are three different rotations (including the identity) that do not alter the graph. These rotations can each be represented as permutations of the vertices, as shown. This can be extended to the capsid, as is done with permutations($permNo, FD$).

turns the FD/axis that represents the input FD/axis, *FD* transformed by permutation *permNo*. This can be simply produced in the algorithm as a variable, populated from reading in a data file that contains all 60 potential permutations (including the identity) that respect icosahedral symmetry. Figure 4.7 shows how permutations can represent symmetric rotations, which is extended to the full icosahedron here. This is key, as the algorithm relies upon icosahedral symmetry to take one capsomer in one FD (or on one axis) and to populate every other FD (or axis) with equivalent capsomers. As the positions are stored as just numbers (which represent FDs and axes), then the various rotations need to be stored as permutations of the integers that represent each FD or axis. Within the algorithm, the permutations variable is looped over for every piece of information that is given in the tile file. If the numbering scheme ever needed to be changed, then that could be achieved by creating a new planar net diagram and using it (or a printed and folded 3D version) to create a new permutations file and then using that new planar net to create the new tile files.

The way that permutations($permNo, FD$) is used can be shown in Algorithm 3

in Appendix B, which reads in the capsomer-capsomer interactions and populates the graph with them. It adds every permutation of the input interaction to the graph. It has a mask, $m(a, b)$ applied to it, so that once an edge is added from a to b, it will not allow this edge to be added again (which would otherwise occur whenever a capsomer sits on a symmetry axis). This method is easily adapted to read in RNA connections and Tile Types however that is not covered here, as these methods are analogous to the population of the capsomer-capsomer graph.

Algorithm 5 in Appendix B describes how all of the variables, functions and algorithms come together to produce a set of arrays and variables that store all of the information on the capsid that is needed to model assembly. For graphs, this information is typically stored as a 2D array, which stores a list of the edges for each vertex, rather than as a sparse adjacency matrix. This allows other information to be stored in similar structures (*e.g.* the ΔG_{bond} values for each edge's respective interaction) in easily accessible and identifiable ways. Any additional information on vertices is stored in 1D arrays.

4.2.1 Discussion Of RNA Graphs

Experimentally determining which RNA graph accurately describes the assembly of a given virus is extremely challenging. Thus, computational simulations are a strong tool to piece together other insights and identify which RNA graph describes the assembly process best. This means that many different RNA graphs are identified and any of them may describe the assembly.

Methods to identify which RNA graphs are most relevant need to be implemented. One way to do this is to calculate the number of RNA arrangements that could exist within a completed capsid. This is covered in Section 5.1.

Another way would be to use the stochastic simulation described in this chapter, to identify the assembly efficiency of each RNA graph. This can be used to select RNA graphs which have a higher yield and to observe how parameters could be changed to increase the yield for these RNA graphs further.

4.3 SIMULATING REACTION DYNAMICS

From the previous section, the simulation knows where the RNA and the capsomers of a capsid can sit but it does not yet know anything about the steps that the

capsid goes through to get from free capsomers to assembled capsids. In this section, the reactions allowed to occur in the simulations are defined, then an algorithm to identify them within a capsid is described, followed by another algorithm that can determine which reaction should fire, using a variation of the Gillespie algorithm covered in Section 2.2. This methodology has been used previously by Dykeman *et al* [19, 18].

4.3.1 Assembly Reactions

Assembly is a process that takes a large number of steps, in which a mixture of free capsomers become fully assembled capsids. To accurately model it, the steps that the assembly is allowed to take must be defined as a set of assembly rules. During assembly, the capsomers will form many different intermediate structures, which can be described as subgraphs of the full capsomer graph. During assembly in ssRNA viruses, the RNA co-assembles with the capsid shell, which results in the RNA tracing out a Hamiltonian path, *i.e.* a path that visits each node in the RNA graph precisely once. The incomplete path traced out by the RNA in an intermediate structure will be termed an intermediate path.

For many ssRNA viruses, the RNA has secondary features, called Packaging Signals (PSs), which can bind to capsomers to aid in the co-assembly process. These PSs are able to help stabilise the intermediate structures during assembly and their presence helps to ensure that the RNA is packaged inside the capsid during assembly. Thus, simulations involving RNA should model PSs. This simulation treats these PSs as if they are beads on a string, formed with a uniform separation, which allows the PSs to bind to capsomers. Whilst in real RNA the separation is not necessarily uniform, we assume that it is here, as this allows all geometric considerations on the RNA to be contained within the RNA graph. If different separations were allowed, then each PS would have a different set of edges available on the RNA graph, based upon the separation between it and its adjacent PSs. This algorithm could be upgraded to facilitate this but this was not implemented. From an analysis perspective, this would add another set of dimensions to the parameter space, by adding the PS-PS length as a parameter for each separation for each RNA graph.

PSs can be made from a wide range of different structures, with different arrangements or different bases present, which means that strength of interactions between PSs and capsomers can vary too. To simplify this for the model, each PS is given a

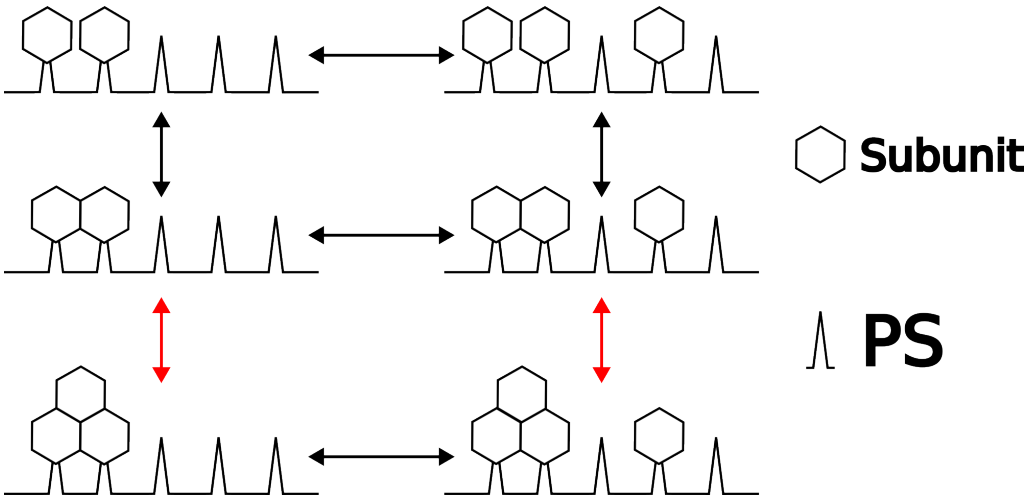

Figure 4.8: Illustration of the reactions that are allowed to occur for capsomers that can bind to PSs. The reactions going right show a capsomer binding to a PS. Downwards, the reactions show two capsomers on adjacent PSs forming an interaction and making an intermediate structure. These reactions are reversible, so the figure also shows the reverse of these reactions in the opposite directions. Once enough of these reactions happen, then the capsid will gain enough capsomers to finish assembly and become a complete capsid. Capsomers that are not bound to a PS are allowed to join the capsid. The red arrow indicate examples where these capsomers bind to the existing intermediate structure.

single parameter value that determines the strength of the bond it can form with a capsomer. However, not all capsomers need to bind to PSs, even in simulations that include the RNA. An example of this is MS2, whose capsomers use the Rhomb tiling shown in Example 3 of Figure 4.3 (but not necessarily the RNA graph). Its capsomer has two different conformers: the AB dimer and the CC dimer, represented by the orange and blue rhombuses, respectively. During assembly, the AB dimers bind to the RNA and the CC dimers do not.

During assembly, the capsomers are able to bind to the PSs. After the capsomers are bound to PSs, they can join the intermediate structure. If a capsomer is bound to a PS, it is only allowed to join the intermediate structure when one of its adjacent PSs is bound to a capsomer that is part of the intermediate capsid structure. These reactions are shown in Figure 4.8 on a small section of RNA.

Capsomers that do not bind to PSs are able to join the structure without being bound to a PS, if they attach to the current intermediate structure, indicated in Figure 4.8 by red arrows. Each of the reactions in the figure are able to reverse, where capsomers leave the intermediate structure. One restriction that has been

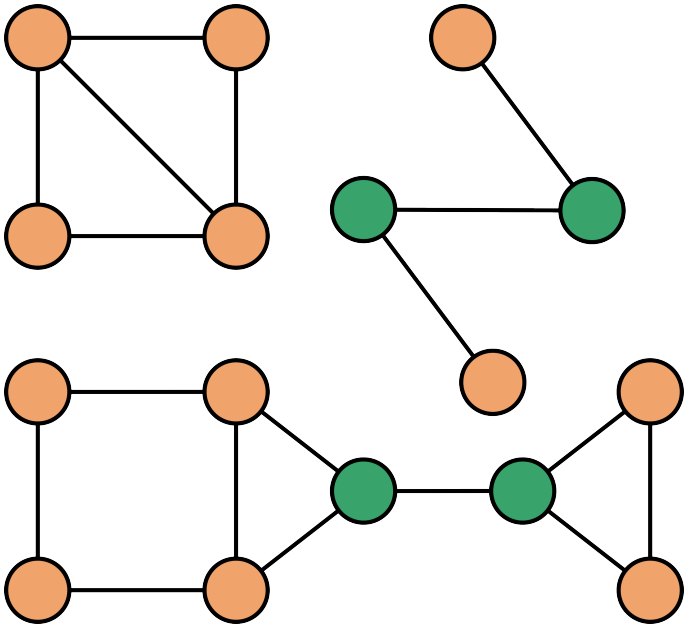

Figure 4.9: This figure shows three example of connected graphs. If some of the vertices in these graphs were removed, the remaining graph would be disconnected. These vertices are called articulation points. The articulation points in these graphs are green and non-articulation points are orange.

placed on this simulation is that no capsomer is allowed to leave if it will cleave the capsid into two parts (not counting the capsomer that leaves). An example of what this would look like is shown in Figure 4.9, where some nodes cannot be removed without disconnecting the graph. Another restriction is that when a capsomer bound to a PS leaves the intermediate structure, this capsomer must be bound at one end of the RNA or the other - none of the PSs in the middle of the RNA's intermediate path can have their capsomers leave the intermediate structure. For example, if there is a path which starts at the first PS and ends at the fifth, then the only PS-bound capsomers that can disassociate from the capsid would be bound to the first and fifth PS: the capsomers bound to the second, third and fourth PS would not be allowed to leave.

Figure 4.10: A flow diagram to summarise the code described in Tarjan's Paper [50].

4.3.2 Finding Possible Capsid Reactions

Now that the capsid reactions have been defined, the algorithm needs to be able to identify which reactions may occur and to calculate their propensities. The algorithm uses a modified version of the SSA algorithm that groups the reactions (each capsid's reactions are one group) and stores the total propensity for each capsid separately. These totals are only recalculated after that capsid reacts [19, 26, 1]. Various methods of grouping reactions are described in more detail in Section 2.2.5.

Unfortunately, a number of reactions depend on the concentration of the capsomers in the system, which changes after every reaction. A workaround for this relies upon the fact that all allowed reactions only depend on one capsomer joining or leaving per reaction, so the propensity for each reaction will at most depend on the concentration of one type of free capsomers. As $\sum_i P_i[n] = [n] \sum_i P_i$, where [n] is the concentration of the n^{th} free capsomer and P_i is the propensity of the i^{th} possible reaction, these families of reactions can be divided further. If there are m types of capsomer tiles in a capsid, then the reactions for this capsid can be further subdivided into $m + 1$ different subgroups. One subgroup will not depend on the concentration of any capsomer and the other m families will each depend on the concentration of one capsomer type. The reactions where a free capsomer of a given type binds to the RNA or joins the capsid intermediate will all depend on the concentration of that capsomer type. The propensities that depend upon concentrations can be stored without multiplying them by the concentration and they will remain accurate until a reaction in the capsid fires. The product of these propensities and the concentration can be calculated later, when the full propensity is needed. This means that an extra binary tree is needed for each different capsomer type.

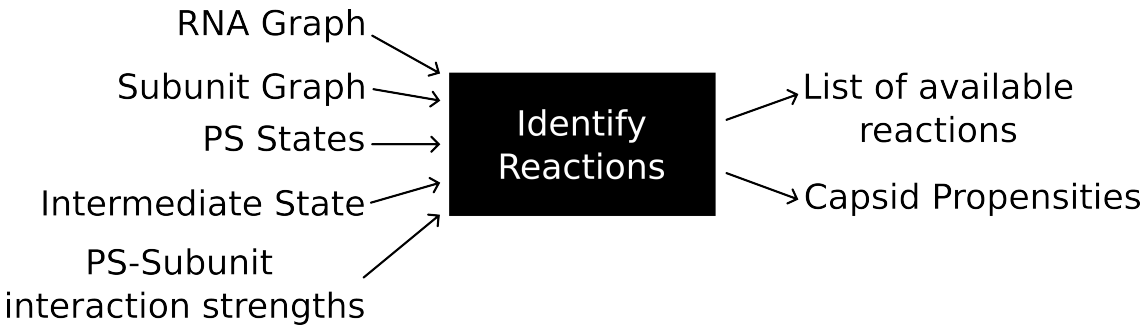

Figure 4.11: A flow diagram to summarise the code described in Section 4.3.2.

During a simulation, the algorithm should store some information about each capsid, including:

1. which capsomer positions on the capsid are occupied
2. which PSs have capsomers bound to them
3. points on the capsid that each PS is bound to (if bound)
4. the strength of each PSs' bond with its CP

Using this information and pre-existing knowledge about the capsid (RNA graph, capsomer graph), it can identify which reactions are allowed to happen and calculate their propensities. Before any reactions can occur, the algorithm should mark some capsomers and positions to prevent them from reacting. The RNA has two ends, the $5'$ end and the $3'$ end. When RNA is produced *in vivo*, it is synthesised from the $5'$ end to the $3'$ end. This is the same direction that the RNA encodes its genetic information in. As the RNA traces out an intermediate path in the capsid intermediate, only the capsomer furthest in the $3'$ or $5'$ direction is allowed to leave. Due to this, any capsomers mid-path are marked so they are not allowed to dissociate. This can be attributed to many reasons. The first is that the combination of RNA-capsomer and capsomer-capsomer interactions gives a stabilising effect, so that it would be more difficult for capsomers to dissociate in this region of the intermediate. Another would be that the RNA has a finite flexibility and if the adjacent two PSs are still bound to the intermediate structure, the central PS will have a limited distance that it can move away from its neighbours.

Also, when a capsid includes capsomers that can associate to and dissociate from the structure without being bound to PSs, then allowing any capsomer of this type to leave could result in the intermediate being cleaved in two, which would complicate the model significantly. Therefore Tarjan's Articulation Point algorithm [50] has been implemented, to mark any capsomer where this is the case and we implement the rule that it cannot dissociate.

The simplest reactions are capsomers joining/leaving a PS. If a PS is bound to a capsomer and this capsomer is not part of the intermediate structure, then this capsomer is allowed to leave. If a PS is unbound, a capsomer may bind to it. The algorithm should search across the RNA to calculate which of these reactions can occur and add them to the relevant propensity total.

When there is no existing intermediate structure, the algorithm should be able to identify when there are two adjacent PSs that are both bound to a capsomer,

as these capsomers can form the first intermediate structure, with them placed in positions that are adjacent in both the RNA and capsomer graph. These reactions rates should be added to the propensity total.

Mid-assembly, the RNA will have a series of its PSs form an intermediate path. At both ends of this path, if the next PS has a capsomer bound to it, then this capsomer may join the intermediate structure. The capsomer at either end of the path is also allowed to leave the structure. Both ends of the RNA should be checked for these reactions and the propensity added to the total.

Lastly, the code should check capsomer positions that do not bind to the RNA (*e.g.* the CC dimer in MS2). If a capsomer can join the structure in one of these positions, its propensity should be added to the total. If one of these positions had a capsomer present and is unmarked, then it is allowed to leave the structure and the propensity should be added to the total too.

Once these reaction propensities have been calculated and summed, they can be stored until they're needed. And each intermediate's propensities will only require an update if a reaction occurs that involves it. An example of an intermediate with these reactions is shown in Figure 4.12.

4.3.3 Non-Capsid Reactions

There is one more reaction that may need to be modelled: the production of a capsomer. In larger viruses, this would be production of proteins, followed by these proteins reacting to form capsomers, though this extra step is neglected in this algorithm. As it is the number of capsomers in the mixture, this must be modelled at a systems level, rather than the capsid level like most reactions listed previously. There are two types of viral assembly scenarios that are considered in this model. The first is *in vitro*, where the protein capsomers and the RNA will be mixed together and allowed to reach equilibrium, with all proteins present at the start of the assembly. The other is *in vivo*, where a virus infects a cell and the cell then gradually produces RNA and proteins, which react to make the assembly capsomers. For *in vitro* simulations, the full number of capsomers and RNA are present at the beginning of the simulation, so this reaction is unnecessary. For *in vivo* simulations, the capsomers will be added gradually over time. Within this simulation, only the capsomers are added gradually and the RNAs are all added at the start of the simulation. The addition of the capsomers is treated as another reaction, with a rate

PS	...	9	10	11	12	13	14	...
CP Bound	...	✗	✓	✓	✓	✓	✓	...

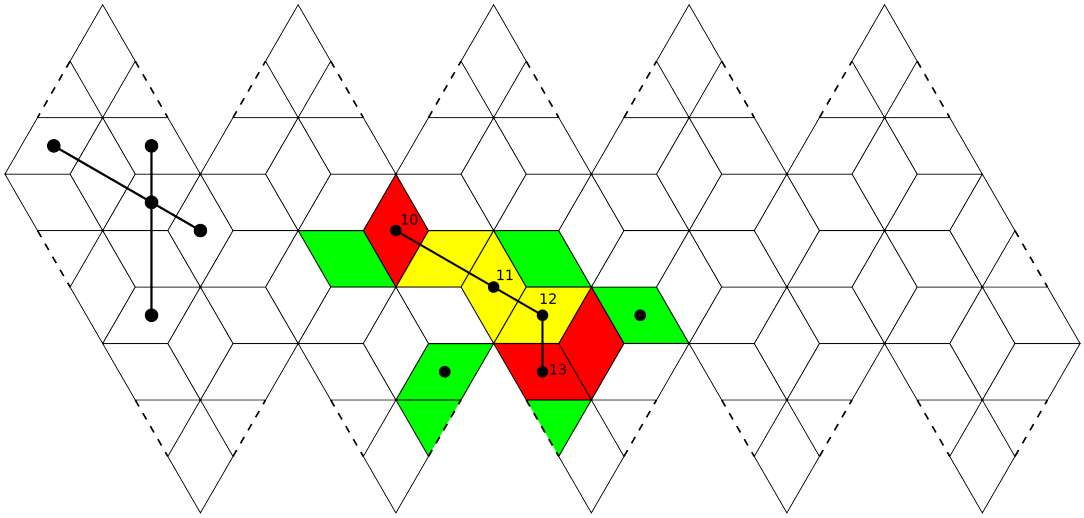

Figure 4.12: An example capsid intermediate using a Rhomb tiling, to demonstrate potential reactions that may occur. Each rhombus is a position that a capsomer may sit, with capsomers that sit on 2-fold axes not being bound to a PS. The RNA graph is shown on the left of the diagram. The table at the top details a small section of the RNA's PSs showing whether they are bound to a CP, with colouring indicating potential reactions. The line with nodes labelled 10-13 indicates where these PS sit on the structure. Green and white shapes indicate that capsomers are not present there, whereas red and yellow shapes have a capsomer present. Green rhombuses and squares indicate places where a capsomer can be added to the intermediate, whether on the RNA or the capsid structure. Green rhombuses with dots are capsomers where a PS-bound capsomer would join. Red rhombuses and squares indicate places where a capsomer can disassociate from the intermediate structure. Yellow rhombuses and squares indicate places where a capsomer is present and is not allowed to leave the intermediate structure. Capsomers can only join/leave the RNA at PSs 9 or 14 respectively (of the PSs shown), as the others are part of the intermediate structure. The dotless green capsomers are capsomers that sit on 2-fold axes that can join the structure. The dotted green capsomers are positions where the capsomer bound to PS 14 could join the structure. There are no dotted green capsomers near to PS 10's capsomer, as PS 9 is not bound to a capsomer. The capsomers bound to PSs 11 and 12 are not able to leave the structure, as they are in the middle of the RNA path. The remaining yellow capsomer is an articulation point, so it is not allowed to leave and cleave the capsid. The capsomers bound to PSs 10 and 13 are able to leave the structure, as they are at the end of the RNA path. The remaining red capsomer is not bound to a PS and so can leave.

Figure 4.13: A flow diagram to summarise the first step of the code described in Section 4.3.4.

that is input alongside other physical quantities. The algorithm has a limit to the number of capsomers that can be created. This limit is normally set to the number of capsomers needed for all RNAs in the system to form completed capsids. When simulations were ran with a larger limit, this did not make a significant change to the yield of the simulations.

This algorithm could be extended to include reactions that change the conformation of a given capsomer as well. An example of this is MS2, where the AB dimer and CC dimer are both made of the same proteins, siting in different conformations. The two dimers can switch between conformations, so a more accurate model would include this reaction. Currently, AB and CC dimers are treated as just being the same capsomer in this model.

4.3.4 Choosing A Reaction

As the algorithm identifies the reactions, it partitions them as the reactions for each individual capsid. There needs to be a way to select which capsid's reactions should be next to fire. As described previously, the algorithm first calculates the total propensities of the reactions for each capsid.

Instead of using the direct method of SSA, other methods give improvements, to more efficiently identify which reaction should be next to fire. The reactions are all allocated into groups (one group per capsid), so that recalculating the total propensity takes fewer steps [19, 18, 20, 17]. In these simulations the total number of RNAs or capsids is usually both large (2000 RNAs when it is present, or up to 60,000 intermediates when the RNA is not present) and fixed, so partial sum tables can be used to store reaction propensities, to help determine which capsid is likely to react next more efficiently.

The queue structure that was used for these simulations is a partial sum table, stored using a binary tree. A binary tree structure is used to store the partial sums

of each capsids' propensities, to allow faster searching. Identification of the next capsid to react will now take $O(\log_2 n)$ rather than $O(n)$. Storing the partial sums in a tree can be done by allowing each node to be the sum of its children nodes. When using Gillespie's Direct Method, the next reaction is usually chosen by finding the smallest value of i such that $\sum_{i'=1}^{i} a_{i'} > ra_0$, where r is a uniform random number in the unit interval and $a_0 = \sum_{i'} a_{i'}$, with reaction j having propensity a_j. With this method, the algorithm searches through each potential reaction in turn. However, when partial sums are stored in a binary tree, half of the potential reactions can be ruled out at each level of the tree.

Once a capsid has undergone a reaction, the propensities of the potential reactions need to be recalculated, as some will no longer be viable and new ones may now be possible. This means one capsid's total propensity has changed, which leads to a need to recalculate the values of the partial sum tree. The only elements on the tree that need to be updated are the node for the capsid that reacted and all of its parent nodes, which speeds up this update.

To determine which capsid should react, Algorithm 2 can be used. It uses the tree structure to efficiently select which capsid should react next and uses partial sums to implement the SSA formulation described by Li and Petzold [36]. This allows the simulation to swiftly determine which reaction happens next and to update only the propensities that have changed.

Within a capsid, the number of potential reactions that could occur changes significantly based upon the current structure of the capsid intermediate. Due to this, it would be challenging to set up a sophisticated structure, such as a tree, to determine which reaction is due to fire next. As a result, the next reaction that fires within a capsid is calculated using the SSA direct method. The number of reactions that could occur in a capsid are typically much smaller than the number of RNAs/capsids in a simulation, so this queue being less efficient is less impactful.

Figure 4.14: A flow diagram to summarise the second step of the code described in Section 4.3.4.

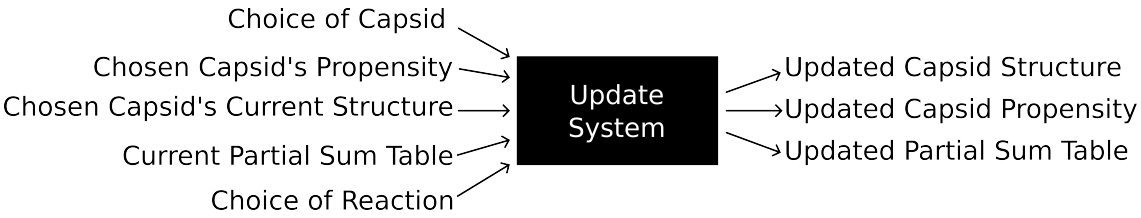

Figure 4.15: A flow diagram to summarise the third step of the code described in Section 4.3.4.

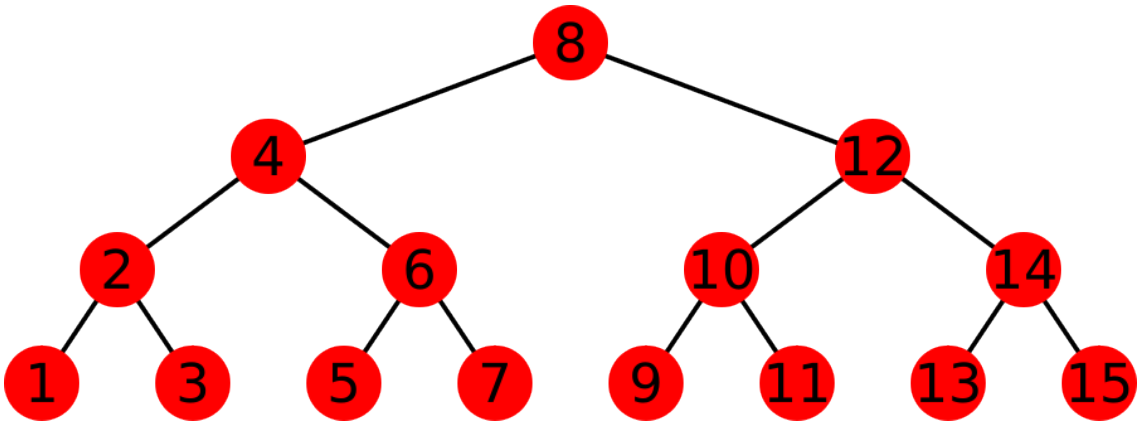

Figure 4.16: An example of a binary tree which can be used to store the partial sums for the system. Each internal node has a parent and two children. For a binary tree that stores partial sums, the parent's value is the sum of the two children's values, *e.g.* $N_8 = N_4 + N_{12} = \sum_{odd\ i}^{15} N_i$, $N_4 = N_2 + N_6 = \sum_{odd\ i}^{7} N_i$, $N_2 = N_1 + N_3$ and N_1 is the sum of reaction 1 and 2's propensities. This can be used as an example of a Gillespie algorithm based on precisely 16 reactions. The tree allows faster searching for the next reaction in a stochastic simulation, as is used in [18].

Once this reaction fires, the capsid's propensity is recalculated and then the partial sum tree is updated. Then the next reaction is selected and the cycle begins again, until the simulation reaches a pre-defined end point.

Algorithm 2 A method to search through a binary tree, to determine which capsid will react next in the simulation

Variables:
$b\_tree[i]$ contains the sum of the propensities of capsids i and $i+1$ for odd i and is a perfect binary tree, using a numbering matching the example in Figure 4.16. Each even element is the sum of its two children
$nsum \leftarrow$ the number of nodes in the tree $+1$
Current node is set to root node, with index $\frac{nsum}{2}$
procedure
  $r \leftarrow$ a uniform random number in unit interval
  $a_t \leftarrow b\_tree[nsum/2] * r$ ▷ The total propensity multiplied by r
  while the current node has children **do**
    $sI \leftarrow$ the index of the smaller indexed child of current node
    if $b\_tree[sI] < a_t$ **then**
      Current node is set to the smaller indexed child of current node
    else
      $a_t \leftarrow a_t - b\_tree[sI]$
      Current node is set to the larger indexed child of current node
    end if
  end while
  $cI \leftarrow$ the index of the current node
  if the cI^{th} capsid's propensity $> a_t$ **then**
    Capsid cI reacts
  else
    $a_t \leftarrow a_t -$ the cI^{th} capsid's propensity
    Capsid $cI + 1$ reacts
  end if
end procedure

— 5 —

Statistical Analysis Of Capsid Assembly Pathways

Following from the model introduced in the previous section, this chapter will introduce a range of different methods used to analyse the results generated through that modelling. This primarily focuses on how the RNA may sit on the inside of the capsid. Initially a simple combinatorics approach was used, giving a sense of the scale of the space of RNA arrangements. Then, tools to analyse the data from the stochastic simulations of assembly described in the previous chapter are presented, which test for any patterns in the arrangements of the RNA inside the capsid. Initially, this is done using a simple approach but when the capsids got larger, more sophisticated analysis tools were developed. Finally, one more tool that was developed is presented. This tool is used to generate informative figures which describe how the RNA may sits inside the capsid at the end of the simulations.

Figure 5.1: A flow diagram to summarise the code described in Section 5.1.

5.1 COMBINATORICAL ANALYSIS OF HAMILTONIAN PATHS FOR GENERAL CAPSIDS

Chapter 4 defines a stochastic simulation to model the assembly of capsids via a Gillespie algorithm, including the way that the RNA sits within the capsid. The available RNA arrangements are defined by the choice of RNA graph, which indicates how two adjacent PSs on the RNA can sit relative to the capsid. Each virus has a large number of available RNA graphs. However, these stochastic algorithms can be slow to run and are unlikely to sample the entire space of possible RNA arrangements (it will only sample the most probable arrangements). Knowing the size of the space that is being sampled from would augment the insight given by the sampling. Thus, a new method was devised, to calculate the number of paths that the RNA can trace out within a given capsid. These paths are usually represented by a Hamiltonian Path.

A Hamiltonian Path is defined as a path on a graph which visits every vertex precisely once. These can be applied to a virus and its RNA, as each PS on the RNA will need to be bound to a capsomer and each capsomer can only be bound to one PS at once. Thus, within a completed capsid, the RNA will have all of its PSs bound to capsomers on the inside of the capsid and will trace out a path on the RNA graph. So the RNA traces out a Hamiltonian path on the capsid's RNA graph.

An algorithm was developed that combinatorically counts all of the Hamiltonian Paths that exist for a given RNA graph. These are read in using the same tile file that was defined in Section 4.2, though it only focuses on the RNA. This process is detailed in Algorithm 6 in Appendix B, which uses recursion to examine the tree of potential paths that could be traced out. This algorithm starts at one end of the RNA (which has *nps* PSs) at one position and builds a path by testing each of the neighbours to this initial position. If they are not disallowed (*e.g.* RNA has already visited it or due to mutual exclusion), they are added to the path and the process starts again with this new, longer path and recurses until the path visits *nps* vertices or there are no available neighbours at the current position.

As the RNA will not necessarily begin its paths at one end of the RNA, ideally the algorithm would not assume this. Equally, it would add a huge computational complexity to test all RNA paths when the path is allowed to go in multiple directions on the RNA and to calculate this starting from every PS along the RNA, then exclude

duplicates. Additionally, allowing the paths to remove capsomers, as would be allowed during the stochastic simulations and which could result in structures that would not be possible otherwise, would exacerbate this issue further. Thus, the algorithm starts its search at one end of the RNA and completely neglects the capsomer-capsomer binding and allows paths to include disconnected capsomer graphs during the combinatorics. Once all PSs are bound (and so the path is completed), the virus likely has all of its *npro* capsomers present (or will if $npro = nps$), so the capsomer graph should not be disconnected at this point. This means all potential paths are counted but that sometimes paths which are valid in the RNA graph but could not occur during the stochastic simulations will also be counted.

This method worked well for smaller values of *nps* but took too much time computationally to run for larger values. This is due to the upper bound on the number of paths being $\approx N^{nps}$ for an RNA graph with N edges on each vertex. As an example, for a graph with $N = 5$, this upper bound goes from $\approx 10^{21}$ with $nps = 30$ to $\approx 10^{42}$ when $nps = 60$, a significant increase.

This algorithm could be improved so that it can deal with these larger paths. In its current form, this algorithm does not do any parallel processing and has efficient memory usage. One method of scaling this would be to run the algorithm so that it will generate all of the intermediate paths of some length *pathLen* and output them all. Then, these paths could be partitioned and a second algorithm would read them in and then continue generating paths from there, allowing many parts of the algorithm to run in parallel. This would still take up a very large computational load but might be feasible.

5.2 ANALYSIS OF RNA PATHS FROM STOCHASTIC SIMULATIONS

For bacteriophage MS2, experimentalists have been able to use cryo-EM to determine roughly where the RNA sits on the inside of the capsid [52]. Due to the techniques they use, this has been averaged over icosahedral symmetry, so it shows where the RNA may sit near the capsid but not the full path. This is effectively an RNA graph, from which all Hamiltonian Paths can be deduced, which was done by Dykeman *et al* [16, 25]. With a small set of rules, they took approximately 40,500 potential paths and narrowed it down to 66 that were consistent with what is known about

the assembly of MS2 and then found the 3 most energetically favourable paths. This allowed comparison with further experimental data, which suggested that the RNA always traces out a specific one of these favourable paths.

From this, there is the suggestion that viruses may tend towards having one path that is highly efficient at assembly and neglect assembly via other pathways. If this is the case, it is necessary to be able to test the stochastic model, to see whether the same effect occurs during assembly simulations where the parameters are tuned for efficient assembly. Analysing the paths, to see how often each path is sampled using the stochastic models may also give insight into why certain RNA graphs will give higher yields than others. If certain RNA graphs give a higher yield and also reliably give the same path, then this is a useful insight and would agree with the prior work on Hamiltonian Paths above. This is valuable to determine computationally, as it is exceedingly hard to get an image of the layout of the asymmetric RNA within a capsid using experimental methods.

As most icosahedral ssRNA viruses use co-assembly processes to ensure the RNA is packaged within the capsid, knowledge of how the RNA achieves this is key to understanding how these viruses assemble. Thus, two methods used to analyse the RNA within completed capsids and the intermediates formed during assembly simulations are described.

5.2.1 Comparison Of Completed Paths

When capsids are fully formed, the surface will be made up of the full number of capsomers for that capsid, in the same positions, so the protein structure of two complete capsids cannot be meaningfully compared. One part of the virus that is not guaranteed to be the same is the path the RNA traces out on the inside of a completed capsid.

In a completed capsid, the RNA traces out a path on the inner surface of the capsid. This path can be labelled from the $5'$ end to the $3'$ end by which capsomers are contacted. This is not necessarily the order that capsomers were added into the

Figure 5.2: A flow diagram to summarise the code described in Section 5.2.1.

capsid, as the assembly could begin anywhere along the length of the RNA. However, if two capsids have the same RNA arrangement, then it is likely that the two of them will have had similar assemblies, so this tells us something about the assembly if one path is more common. Initially, analysis focused upon the paths that were traced out in completed capsids. Only paths generated from stochastic simulations were considered. These paths were stored as a series of positions that the PSs were bound to.

An intuitive way to analyse these paths would be to assign each RNA connection a different number (*e.g.* anticlockwise about a 5-fold axis is 1, clockwise is 2, etc...) and to store the path as a series of these numbers. This works well for viruses that have capsomers that exclusively sit within FDs, such as the example shown in Figure 5.3a, as you can always uniquely define each connection via rotations about nearby symmetry axis, due to the asymmetry of the FD. Unfortunately, this is not the case in capsids where capsomers sit on symmetry axes, such as the one shown in Figure 5.3b. As these capsomers sit on symmetry axes, then the RNA connections cannot be uniquely defined based off their nearby symmetry axes, as there are multiple axes that are indistinguishable due to the symmetry. One workaround for this would be to define a new map to define each move for each position, based on where has been visited most recently by the RNA, as shown in Figure 5.3c. This could be generalised so that it works alongside the tools already presented which produce a map of the capsid. However, it would be challenging to produce requiring knowledge of which rotation about which axis is represented by each of the permutations and also additional user input to generate this map for every capsid, which is what the tool was designed to avoid, so this method was not pursued.

Thus, another method is proposed. It reads in each of the n paths sampled by the stochastic simulation and creates an array containing the $60n$ paths and permutations of these paths, as is described in Algorithm 8 in Appendix B. These permutations represent each of the 60 rotations that leave a capsid invariant and so gives all possible perspectives on how the path traced out by the RNA could sit.

Then, the algorithm takes the first sampled path and compares each following sampled path and all of their permutations to see if they match. When the current path matches a new path or permuted path, a counter is incremented and then the new path and its permutations are marked, so they will not be counted again.

Once this is complete for the first path, it is repeated for the next non-marked path. This is repeated until all paths are marked, with up to n different paths and a

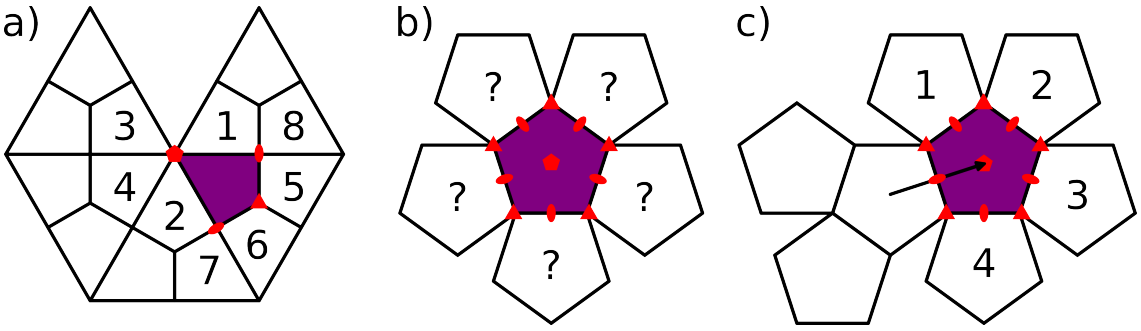

Figure 5.3: Sections of a planar net of (a) an icosahedron and (b),(c) a dodecahedron. They can be used to indicate why generalising a numbering scheme for the edges of an RNA graph can be challenging. Each capsomer is in the shape of (a) a kite or (b),(c) a pentagon, with the purple capsomer representing a capsomer that is bound to a PS and nearby capsomers that adjacent PSs along the RNA could bind to are given identifiers. The red pentagons, triangle and ovals represent 5-fold, 3-fold and 2-fold axes on the structure. In (a), the capsomers each sit within their own FD and any other position (with the eight represented by rotations about an axis that overlaps with the purple FD labelled here) can be uniquely described, relative to the current capsomer by rotations about a nearby axis. In (b), a capsomer that sits on a 5-fold symmetry axis is centred. Due to it sitting on an axis, it is not possible to describe the nearby capsomers uniquely by rotations, as was possible in (a), because all neighbouring axes are rotationally equivalent to each other. For (c), a potential workaround to the issue in (b) is shown. By orienting the path using the previous edge in the path, the nearby capsomers can be uniquely described by rotations, as the previous edge breaks the symmetry. This would require knowledge of which permutation represents which rotation about which axis, which would be time consuming to identify.

total number of times each path was sampled. These paths are sorted based on their totals and every path that occurs at least c times for a chosen value of c is output to be investigated. This is covered in Algorithm 9 in Appendix B.

By using the whole path and its permutations instead of assigning labels to each RNA connection, the algorithm is guaranteed to be able to work. As the simulated capsids are generated using these permutations, then the permutations can always be applied to paths/positions within the capsid, regardless of whether capsomers sit in FDs or or on symmetry axes. It also requires no additional user input other than the tile file to do this analysis.

This method requires more comparisons and is less efficient than the previously proposed method but was sufficiently fast when applied to collections of $n \approx 10,000$ paths sampled using stochastic assembly. This method does result in outputs that

are harder to interpret without visualisation techniques, when compared to the previous method. However, it makes it easier to produce imaging tools, to view the paths in 3D (covered in Section 5.3).

This method was most useful for STNV (discussed in Chapter 6), where there was a large degree of similarity between paths but it can also be used on paths made from any tile. For larger viruses, this similarity between paths was not observed, so other analysis was necessary to see how the simulations were running.

Initially, to try and get more insight into larger viruses, this analysis was performed for smaller sections of the completed paths (*e.g.* testing how often in the paths does the RNA visit all 5 positions about a 5-fold axis in clockwise order). This initially appeared to give new insights but it relied upon viewing these sections in isolation from the other capsomers present in the completed structure due to other sections of the RNA and this would often lead to these sections being disconnected on the capsomer graph. The simulations that resulted in these paths had gone through every step necessary to form these capsids and the intermediates were always connected on the capsomer graph. However, this analysis was viewing one screenshot of the completed capsid and only viewing small sections of its RNA. Therefore it did not give useful insights. These limitations lead to the method covered in the next section.

5.2.2 Analysis Of Intermediate Structures During Assembly

Looking only at the final path that the RNA traced out gave some insight into the assembly of STNV. However, it was less appropriate for larger capsids. Being able to identify key moments during the assembly of a capsid is important to understand the processes that lead to efficient assembly. This leads to the conclusion that data needed to be generated that records the reactions a capsid takes during assembly,

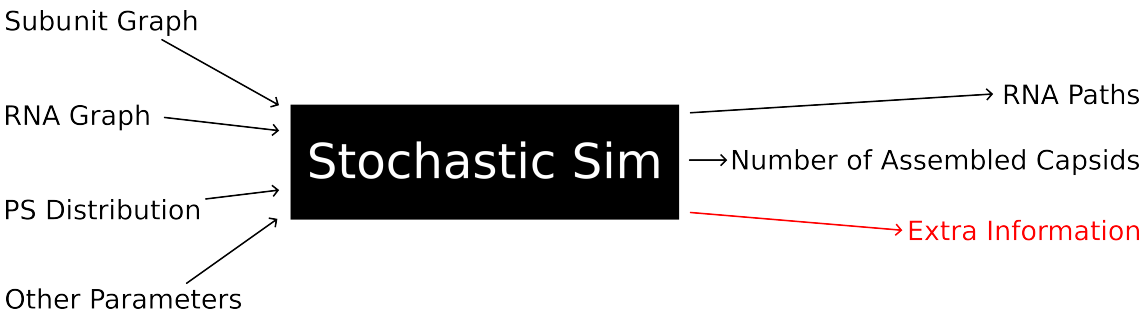

Figure 5.4: A flow diagram to summarise the changes to the code from Chapter 4 that are described in Section 5.2.2, with the additional output labelled in red.

rather than just the end state of the capsid. If efficient assembly of one virus has markers that can be sought out in other viruses, then this would improve our understanding of the assembly process.

The stochastic simulation was modified so that it would store certain information about the capsid for each RNA during its assembly. This information had to be fairly simple, as if it was too large, storing it would impact the simulations speed and use too much memory but also needed to be complex enough to give insight into the capsid's assembly. The following information was gathered:

1. The total number of reactions that occurred, where a reaction is defined as any event where a capsomer joins/leaves the existing intermediate structure and where at least one capsomer-capsomer interaction is formed. Capsomers binding/unbinding from a PS are not included
2. What order the capsomers were added/removed in
3. Which PSs on the RNA were bound to the first two capsomers that made a capsomer-capsomer interaction
4. Which direction along the RNA ($3'$ or $5'$) was a capsomer added/removed from
5. How many capsomers are present post-reaction
6. How many capsomer-capsomer interactions were formed/broken

After this was implemented, a cap was added, so that the algorithm would only record the first 3000 reactions that occurred for each RNA, as simulations typically involved capsids that had < 100 capsomers and it was assumed this would be sufficiently many reactions for capsids to complete assembly. Unfortunately, almost every simulation ran at that point had most of the RNAs involved in over 3000 reactions before completion. From inspecting the data, most of these were due to a capsomer joining the structure and then immediately leaving the structure as the next reaction (or leaving then rejoining). These are two sequential reactions with no net change to the capsid, which can happen very quickly, giving an excess of unhelpful data. This is not unexpected, as it is modelled using a Gillespie algorithm, which are prone to this but a fix was needed. As a result, a filter was added to the algorithm so that if two subsequent reactions were the addition/removal of the same capsomer at the same end of the RNA, it would be removed from the record. This filter does not remove all sequences which give no net result. For example it does not cover a sequence of 4 reactions, two on each end of the RNA, whilst alternating

which end reacts. In theory, it could be worked around by using a marker for each end of the RNA and looking back further. However, this could lead to the data appearing inconsistent. For example, in a capsid where some capsomer A is added at the $5'$ end, then some capsomer B joins on the $3'$ end adjacent to A and then capsomer A leaves, removing the two reactions including A would lead to the number of capsomers jumping by 2 during the reaction that added B (which is not allowed in the stochastic simulations). The number of interactions formed in this way would appear inconsistent as the removal will have reduced the number by 1 more than when it was formed and the rest of the data would be impacted in ways it would be challenging to fix.

For viruses where every capsomer joins whilst bound to a PS, this simple filter was enough to ensure that the number of reactions stored was more manageable. When viruses allow capsomers to join the structure without being bound to a PS, this filter is not enough, as many different weakly interacting capsomers are able to join in any order and so the number of reactions was not reduced by as much (though there was still some effect). No further filters were applied.

This information can be used in two different ways. The simplest method is just to graph them, which can show many intuitive results, for example that it is much more common for a reaction to break 1 interaction than 2. Or less intuitive ones, such that the capsid intermediate can often get stuck for a while at a small size, doing many forward and backwards reactions but after the capsid intermediate reaches a certain minimal size, it almost immediately proceeds to completion, which is consistent with nucleation theory.

The other way to use this information relies upon the fact that it can be used to track the exact series of reactions that each capsid underwent to go from free capsomers to assembled capsid. Unfortunately, trying to get a conclusion from such a large dataset would be challenging, so a smaller section of it was sectioned out and analysed. Using the steps that occurred, an algorithm was developed that followed the instructions and recorded the latest intermediate of n capsomers for $n \in [2, npro]$ for each RNA, where $npro$ is the number of capsomers in one complete capsid. So, for every completed capsid, there would be $npro - 1$ latest structures. The latest structure was chosen as it is the last time the intermediate is that size. If there are a lot of forwards and backwards reactions, then the intermediate which reacts forwards without going backwards after, is the structure that is interesting to look at and compare to other RNAs, as this is the one that proceeds towards a stable capsid.

Once these structures were obtained, it was necessary to compare them to each other to see if there was any correlations in the structures formed. Initially, three criteria were chosen to indicate whether two structures were the same:

1. The same (or rotationally equivalent) capsomers are present in both structures
2. The RNA traces out the same path along the capsomers in both structures
3. The same PSs are bound to the same position in the path in both structures

These were chosen to find any similarities between the assemblies, with the third criterion being focused on identifying how often the same nucleation sites are used. However, this turned out to be too strict and so was dropped shortly after.

The second criterion was also useful but unfortunately, for larger capsids, the space of possible paths is very large, so this criteria needed to be neglected too in many cases. Which leaves the information on where capsomers are added, which can show a lot about what the capsid is doing, as this depends on both the capsid structure and also the RNA graph used in that simulation. So the algorithm eventually generates a tally of how many times each structure is the most recent structure of a given size for that RNA from a simulation. This is done using a similar method to the one discussed in Section 5.2.1, where each permutation is applied to the structures followed by comparison to the other alternative structures for each intermediate size.

There is the caveat that this method only generates useful results when $npro = nps$ due to capsomers that are not bound to PSs being prone to join/leave the capsid structure more frequently than those bound to PSs (covered in more detail in Section 3.2). As a result, the latest intermediate is more likely to be due to fluctuations than significant steps in the assembly.

5.3 IMAGING OF PATHS

Being able to view the RNA paths generated by the stochastic simulations is a useful way to compare them and to see how the capsid assembles. Additionally, it can also be useful to view the capsomer and RNA graphs, to compare across capsids. Thus, a process is needed to generate 3D images of the graphs that make up the capsid. This is done in tandem with the makemap algorithm from Algorithm 5 in Appendix B.

It uses the same tile file used previously and one new file with the coordinates of each distinct capsomer's binding site (in the case of RNA paths) or each distinct

capsomer's centre of mass, alongside their PN and FD. The rotations are applied by knowing a series of rotations which transform the initial FD/axis into the desired FD/axis. Due to this, the input FD must be 1, 61, 91 or 111, rather than any FD being allowed, like in the tile file. These coordinates do not need to be normalised, so if a specific virus is being modelled, then the coordinates of its binding sites can be used and the output file can be viewed alongside the PDB file for the virus and so indicate interesting features on the capsid with this.

The algorithm takes these inputs then, using a predefined series of rotations about some of the nearest symmetry axes, generates the coordinates for each copy of these input capsomers. These series of rotations are stored as anti/clockwise rotations about the 5-fold and 3-fold axes that sit closest to FD 1. The normalised axes' coordinates are defined as: $\mathbf{ax}_5 = \frac{1}{2+\phi}(0, 1, \phi)$ and $\mathbf{ax}_3 = \frac{1}{\sqrt{3}}(1, 1, 1)$, where $\phi = \frac{1+\sqrt{5}}{2}$, the golden ratio. The region of FD 1 can be defined as the kite on the surface of the unit sphere with vertices $\mathbf{ax}_5$, $\mathbf{ax}_2$, $\mathbf{ax}_3$ and $\mathbf{ax}_{2'}$ and any vector (when normalised) that sits in this space is within FD 1. The positions of the 2-fold axes are: $\mathbf{ax}_2 = \frac{1}{2}(1, \frac{1}{\phi}, \phi)$, which represents 2-fold axis 1 and $\mathbf{ax}_{2'} = \frac{1}{2}(\frac{1}{\phi}, \phi, 1)$, which represents 2-fold axis 3, both shown in Figure 4.6.

For a general axis $\mathbf{u} = (u_x, u_y, u_z)$, the general rotation can be written as:

$$R(\mathbf{u}, \theta) = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & k \end{pmatrix} \tag{5.1}$$

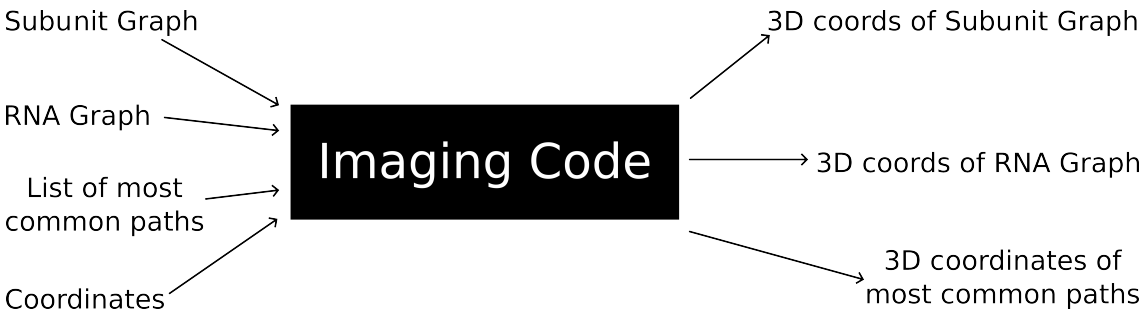

Figure 5.5: A flow diagram to summarise the code described in Section 5.3.

where

$$a = cos\theta + u_x^2(1 - cos\theta) \tag{5.2a}$$
$$b = u_x u_y(1 - cos\theta) - u_z sin\theta \tag{5.2b}$$
$$c = u_x u_z(1 - cos\theta) + u_y sin\theta \tag{5.2c}$$
$$d = u_y u_x(1 - cos\theta) + u_z sin\theta \tag{5.2d}$$
$$e = cos\theta + u_y^2(1 - cos\theta) \tag{5.2e}$$
$$f = u_y u_z(1 - cos\theta) - u_x sin\theta \tag{5.2f}$$
$$g = u_z u_x(1 - cos\theta) - u_y sin\theta \tag{5.2g}$$
$$h = u_z u_y(1 - cos\theta) + u_x sin\theta \tag{5.2h}$$
$$k = cos\theta + u_z^2(1 - cos\theta). \tag{5.2i}$$

So the rotation of a vector $\mathbf{v}$ about axis $\mathbf{u}$ by angle θ is represented by:

$$\mathbf{v}_{rot} = R(\mathbf{u}, \theta)\ \mathbf{v}. \tag{5.3}$$

A function to perform this rotation is given in Algorithm 10 in Appendix B, which allows 4 combinations of θ and $\mathbf{u}$ to be applied to input $\mathbf{v}$.

This function is used alongside the set of rotations the algorithm reads in within Algorithm 11 in Appendix B. This algorithm uses the list of rotations that can map $1 \mapsto [1, 60]$, $61 \mapsto [61, 90]$, $91 \mapsto [91, 110]$ or $111 \mapsto [111, 122]$ to take the input coordinates, then take them through all necessary rotations and produce the coordinates for all positions within the capsid.

There are two useful ways to output these positions. The first is to use it alongside the capsomer and RNA graphs, as defined in Section 4.2, to generate a 3D projection of the capsomer graph or the RNA graph. The way to do this is shown in Algorithm 12 in Appendix B, so that these graphs can be produced and compared. Also, it can be useful as a backdrop for the second way to output these positions. This is to use a path that has been generated during stochastic assembly (typically one that is frequently found during assembly) and to make a graph that traces out a path between all the points on the inside of the capsid that the RNA binds to. These are all done as straight lines between binding sites, so do not represent exactly where the RNA sits, as it is more flexible than that but it does give a sense of the sequence of binding sites visited along the linear genomic sequence. The images produced by these techniques will be seen in later chapters.

— 6 —

Modelling STNV, A $T = 1$ Virus

This chapter covers the results from modelling STNV using methods presented in previous chapters. Some of these results are presented in a paper which was published during this PhD [32].

The Satellite Tobacco Necrosis Virus (STNV) is a small, $T = 1$ satellite plant virus, which means that its capsid shell is made up of only 60 proteins. As a satellite virus, its genome is so small that it does not encode all of the necessary instructions for it to replicate and thus relies upon a helper virus, in this case Tobacco Necrosis Virus (TNV), to co-infect the cell for it to be able to replicate. Its genome is made up from only 1239 nucleotides and is stored as single stranded RNA (ssRNA). STNV infects plants, including tobacco, so has economic importance. Additionally, it is a good model system, as it only consists of ssRNA and proteins and has been researched well using experimental techniques [34, 35, 42, 43, 6, 22].

6.1 GEOMETRY OF STNV CAPSID

As detailed in Chapter 4, to model STNV both a capsomer graph (which gives information on which capsomers are adjacent to each other) and RNA graphs (which gives information on where the RNA can sit inside the capsid) need to be made. To construct the capsomer interaction graph, experimental date was used: cryo-EM data on the protein shell of STNV that has been imaged to a resolution of 1.45Å, as shown in Figure 6.1 [34], in addition to some data on how the RNA might sit inside. This can be used to generate a 2D net of where each of the proteins sit relative to each other, as is given in Figure 6.2.

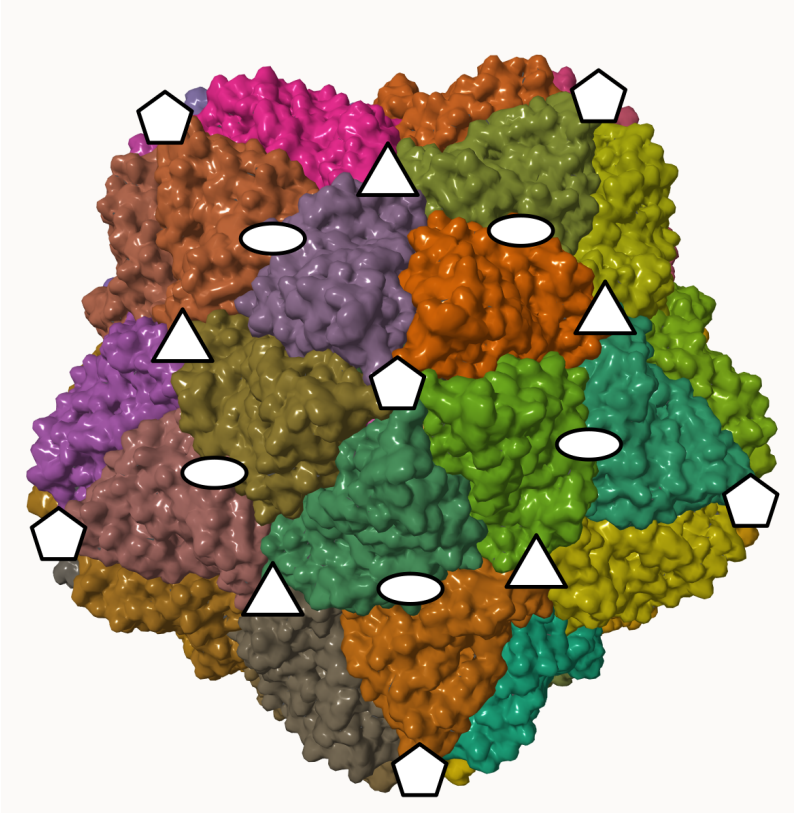

Figure 6.1: An image of the external surface of STNV, produced using ViperDB's imaging tool (PDB ID: 4V4M [41]). Each protein capsomer is given a different colour, to indicate where each protein ends. At the centre of this figure is a 5-fold axis (indicated by a pentagon), with 5 proteins sitting around it, which have a large contact area between proteins that sit about that axis. There are five 3-fold axes visible (shown with triangles), with low contact area between proteins in this area. There are also five 2-fold axes (labelled with ovals) present, with a more noteworthy contact area across neighbouring proteins beside this axis.

Each protein has a large contact area with the adjacent proteins about the 5-fold and 2-fold axes and a small contact area with the proteins about the 3-fold axis. One feature worthy of note is the N-terminal alpha-helices that are near to the 3-fold axis on each protein, shown as dots within the CP in Figure 6.2. These are positively charged (due to lysine and arginine) and so apply a repulsive force to the other alpha-helices from other proteins located around the three-fold. The negatively charged RNA is believed to make contact with the alpha-helices and thus reduces this repulsive interaction, enabling the proteins to form a capsid [22]. This interaction means that it is harder for the capsid to assemble in the absence of the RNA. So, in the simulation, the proteins are considered to make 3 CP-CP bonds and

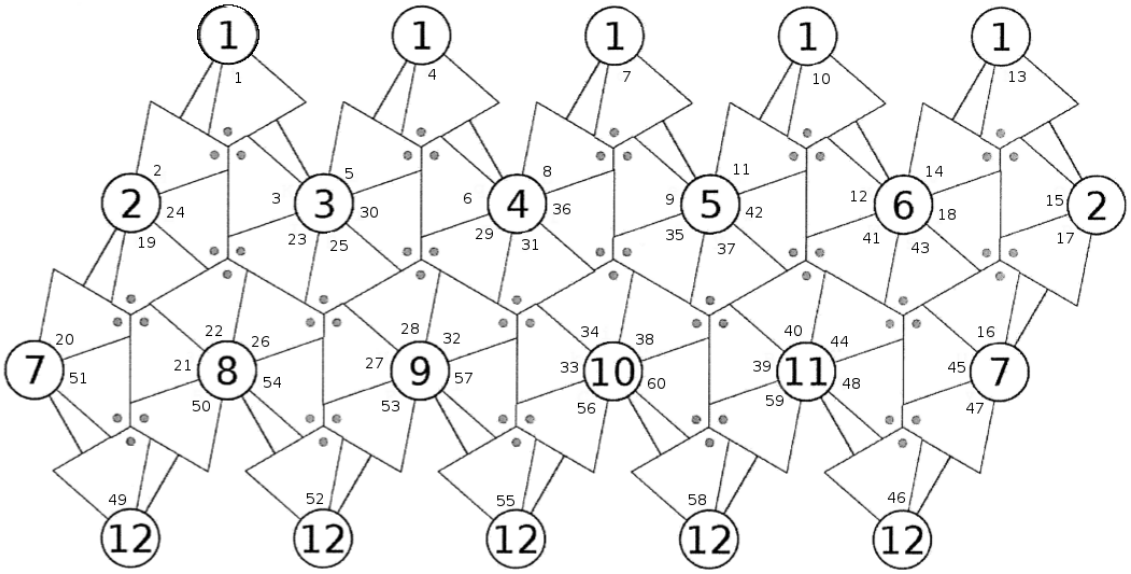

Figure 6.2: A 2D net that represents the CPs (shown as rhomboids) from an exterior view of the capsid of STNV, numbered using the scheme described in Section 4.2 (adapted from [24]). Each 5-fold axis is labelled, to indicate where overlaps occur in the net. The dot in each CP represents the alpha-helix's position in that CP.

the 3-fold axis is considered to contribute no energy with these interactions. This is an appropriate simplification, because any binding energy in this area would be small, compared with the repulsive interaction of the alpha-helices and can thus be neglected.

6.2 POSITION OF RNA CONTACTS WITHIN CAPSID SHELL

As discussed in Chapter 4, the layout of the RNA's PSs and the edges available on the RNA graph will have a substantive effect on capsid assembly.

We next consider the RNA. Experimentalists used SELEX to identify potential PSs in STNV's genome. One example found was the B3 hairpin, which triggered reassembly of STNV VLPs when present [6]. The B3 hairpin is shown in Figure 6.3. Capsids assembled in presence of many copies of just the B3 hairpin were then imaged. After the icosahedral averaging of the imaging was applied, the B3 hairpin was observed in 60 positions across the capsid, however Lane *et al* noted that the electron density was consistent with only 30 hairpins [35].

When the positioning of the RNA helix is observed, the helix occupies a space leading from the N-terminal alpha helix on the CP to the 2-fold axis of the capsid.

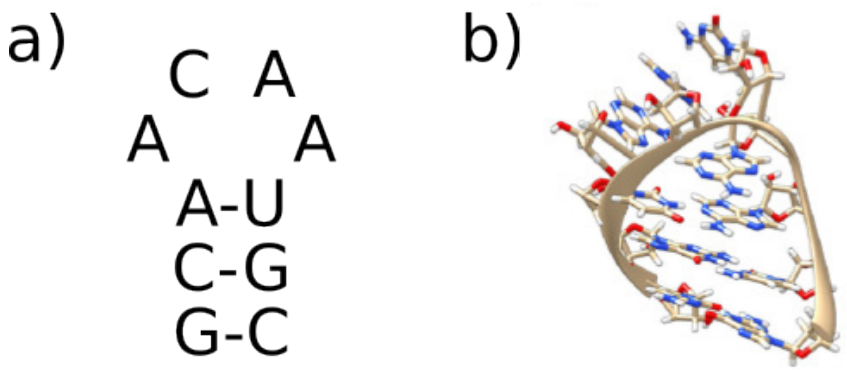

Figure 6.3: The B3 Aptamer, which is an example of a PS for STNV. (a) The sequence of bases that forms the aptamer. (b) The structure of the B3 aptamer.

Figure 6.4, shows how close two of these helices bound to the interior of the capsid sit and it has been postulated [6, 22] that the apical loops would sit in a way that would cause steric clashes.

These postulated steric clashes suggest that the RNA would be unable to bind to both of these CPs in the capsid. This is supported by the experimental evidence suggesting only half of the CPs had RNA bound to them in the above experiment. Using both of these observations, as shown in Figure 6.4, only one of the two purple CPs are able to bind to the RNA. In other words, RNA binding both of these sites is mutually exclusive. Thus, during assembly simulations, the RNA is only able to be bound to one of them at any point. Whenever the RNA binds to a given node, the mutually exclusive partner node is marked so that the RNA cannot bind to it. This marking also allows unbound proteins to join the intermediate structure at this node too during the simulations. Only allowing a free capsomer to join at a position after its mutually exclusive partner is bound to a PS can be justified in two ways. One is that in some viruses, the capsomer-capsomer interactions may be too weak to allow the unbound capsomer to join the structure and the RNA sitting on the nearby 2-fold axis will provide a stabilising interaction. Another more practical reason is that capsomers not bound to a PS tend to be less stable than capsomers that are bound to a PS, so including all of the reactions initially would cause a larger number of binding/unbinding reactions that will slow down the assembly of the capsid, so it is excluded.

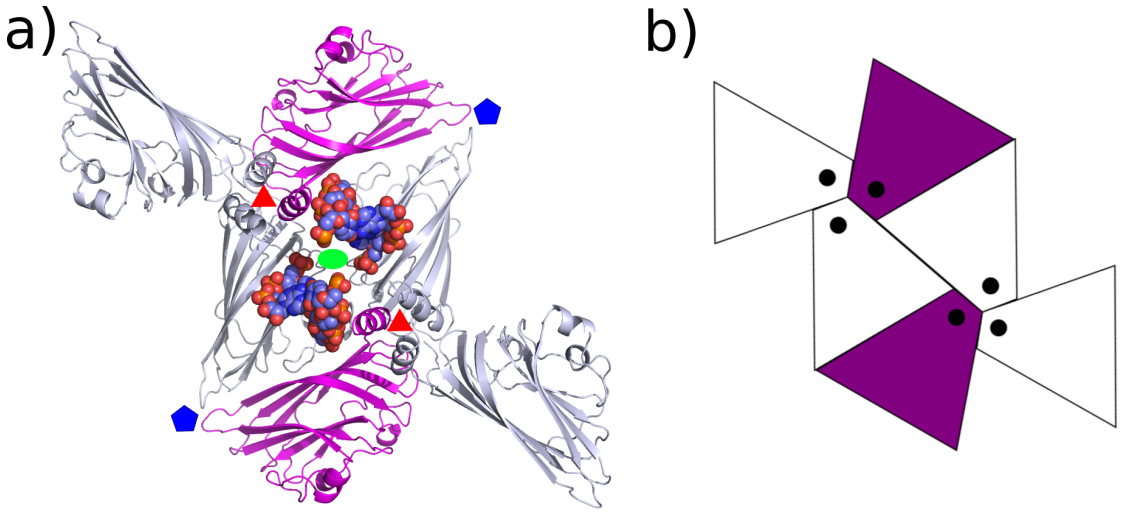

Figure 6.4: (a) Interior surface of the structure of the STNV capsid in complex with the B3 Aptamer. The position of the bound aptamer is shown for the two purple CPs. Not shown is the apical loop, which is theorised to sit in a position that would clash with the other aptamer. The 5-fold, 3-fold and 2-fold axes are labelled with pentagons, triangles and ovals. (b) A planar representation of the mutual exclusion rule, viewed from the inside, where the two purple CPs are mutually exclusive and only one can be bound to a PS. The purple CPs correspond to the purple CPs in (a). This mutual exclusion can be mapped onto the whole capsid, so that each CP has a mutually exclusive partner.

6.3 CHOICE OF RNA GRAPHS

At this point, we have the capsomer graph, which determines where STNV's CPs sit relative to each other. Next, the graph that indicates where the RNA can sit is needed. It is hard to fully experimentally determine where the RNA that connects between adjacent PSs might sit on the interior of the capsid. So a set of potential connections between adjacent PSs, which will be the edges in the RNA graph with vertices representing PS-CP contacts, must be proposed and tested. The proposed edges for STNV's RNA graph each connect Figure 6.5's node 0 to one of the other 9 nodes that can be reached by rotations about the nearest symmetry axes. These edges determine in which directions the RNA can move and how it can trace out a path on the interior of the capsid. These connections are subject to the rotational symmetries of the capsid that are shown in Figure 6.5, resulting in Table 6.1. They have been assigned numbers to indicate which edges (the edge connecting 0 to i for $i \in [1, 9]$ is labelled as edge i) are present in the RNA graph by listing the edges that

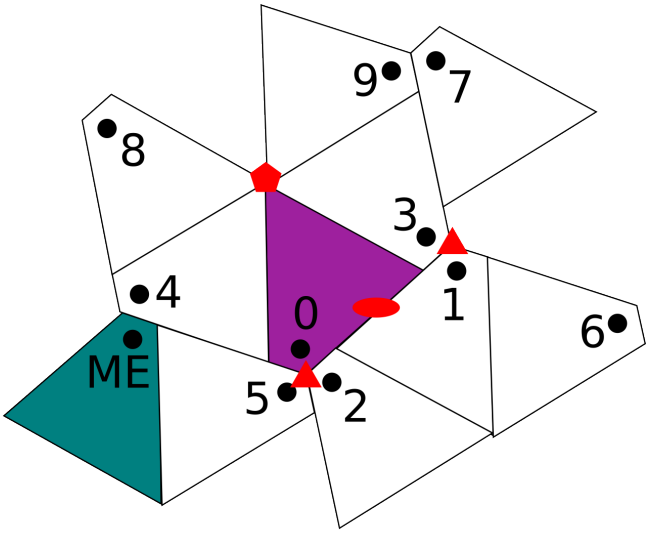

Figure 6.5: The 9 permitted vertex connections for STNV's RNA graph, as viewed from the interior, adapted from predictions in James Garaetts' PhD Thesis [24]. If the RNA has a PS bound to the purple protein, then these edges ($[0, i]$ for $i \in [1, 9]$) represent the protein positions that a PS adjacent on the RNA could be bound to. Embedding a selection of these edges onto Figure 6.2 and drawing edges between the alpha helices will result in the RNA graph for that assembly scenario. The nearest axes are labelled with a pentagon, triangle and oval representing the 5-fold, 3-fold and 2-fold axes respectively. For convenience, the mutually exclusive partner to the purple protein is shown in teal and labelled as "ME". For the capsomer graph, the purple CP is considered to be adjacent to proteins 1, 3 and 4.

are present.

The RNA graphs for STNV are generated by using a subset of these 9 edges, with examples drawn in Figure 6.6. Within this RNA graph, all edges are bidirectional. An example of this is edges 3 and 4, which can be represented as rotations by $\frac{2\pi}{5}$ about the nearest 5-fold axis in the clockwise and anti-clockwise directions. When considering these edges as rotations, the edges are paired with their inverse. So, edge 2 is paired with 5, 6 with 7 and 8 with 9. As it is across a 2-fold symmetry axis and thus self-inverse, edge 1 does not have a partner, so can be included on its own.

Thus, RNA graphs will have degrees between 4 and 9, corresponding to a subset of these 9 edges being allowed as connections. Subsets with 3 or fewer edges are not able to assemble, thus were neglected.

Choices of edges will be listed with the edges present within curly brackets, *e.g.* $\{2345\}$. This will contain the edges that lead from protein 0 to each of the 4 listed

Edge	Rotation
$0 \to 1$	Rotation by π about nearest 2-fold axis
$0 \to 2$	Clockwise rotation by $\frac{2\pi}{3}$ about nearest 3-fold axis
$0 \to 3$	Anti-clockwise rotation by $\frac{2\pi}{5}$ about nearest 5-fold axis
$0 \to 4$	Clockwise rotation by $\frac{2\pi}{5}$ about nearest 5-fold axis
$0 \to 5$	Anti-clockwise rotation by $\frac{2\pi}{3}$ about nearest 3-fold axis
$0 \to 6$	Anti-clockwise rotation by $\frac{2\pi}{3}$ about second nearest 3-fold axis
$0 \to 7$	Clockwise rotation by $\frac{2\pi}{3}$ about second nearest 3-fold axis
$0 \to 8$	Clockwise rotation by $\frac{4\pi}{5}$ about nearest 5-fold axis
$0 \to 9$	Anti-clockwise rotation by $\frac{4\pi}{5}$ about nearest 5-fold axis

Table 6.1: The edges, represented as vertex pairs, from Figure 6.5, describe rotations from the purple protein to the protein that is connected via this edge. Each edge has an inverse edge, *i.e.* an edge whose rotation corresponds to the inverse of the initial edge's rotation, pairing 1 with itself, 2 with 5, 3 with 4, 6 with 7 and 8 with 9.

vertices, as shown in Figure 6.5. A visualisation of this RNA graph is given in Figure 6.6a. A full list of potential choices can be found in Table 6.2, which also includes the number of edges per vertex alongside stochastic and combinatorical data. One choice of edges that is absent from this table is $\{3489\}$, which only contains edges that go about the 5-fold axis, so it would not be able to form a completed path and has thus been excluded.

6.4 HAMILTONIAN PATH COMBINATORICS IN STNV

There are three key questions to consider regarding the assembly of STNV's capsid. What is the assembly path that the virus takes? What is the organisation of the RNA packaged inside of the capsid? Does the packaged RNA have a single organisation or does it have many?

One way to categorise the way that a capsid assembles is to observe the paths traced out by the RNA within a completed capsid. These can be sampled using stochastic simulations, but knowledge of the size of the space being sampled is also useful.

As covered in Section 5.1, a depth first search algorithm can be implemented to find the number of Hamiltonian paths that can exist on the RNA graph. Normally, a virus would have the RNA trace out a Hamiltonian path, where each vertex on its RNA graph is visited exactly once but this is not possible in STNV, as only 30 out

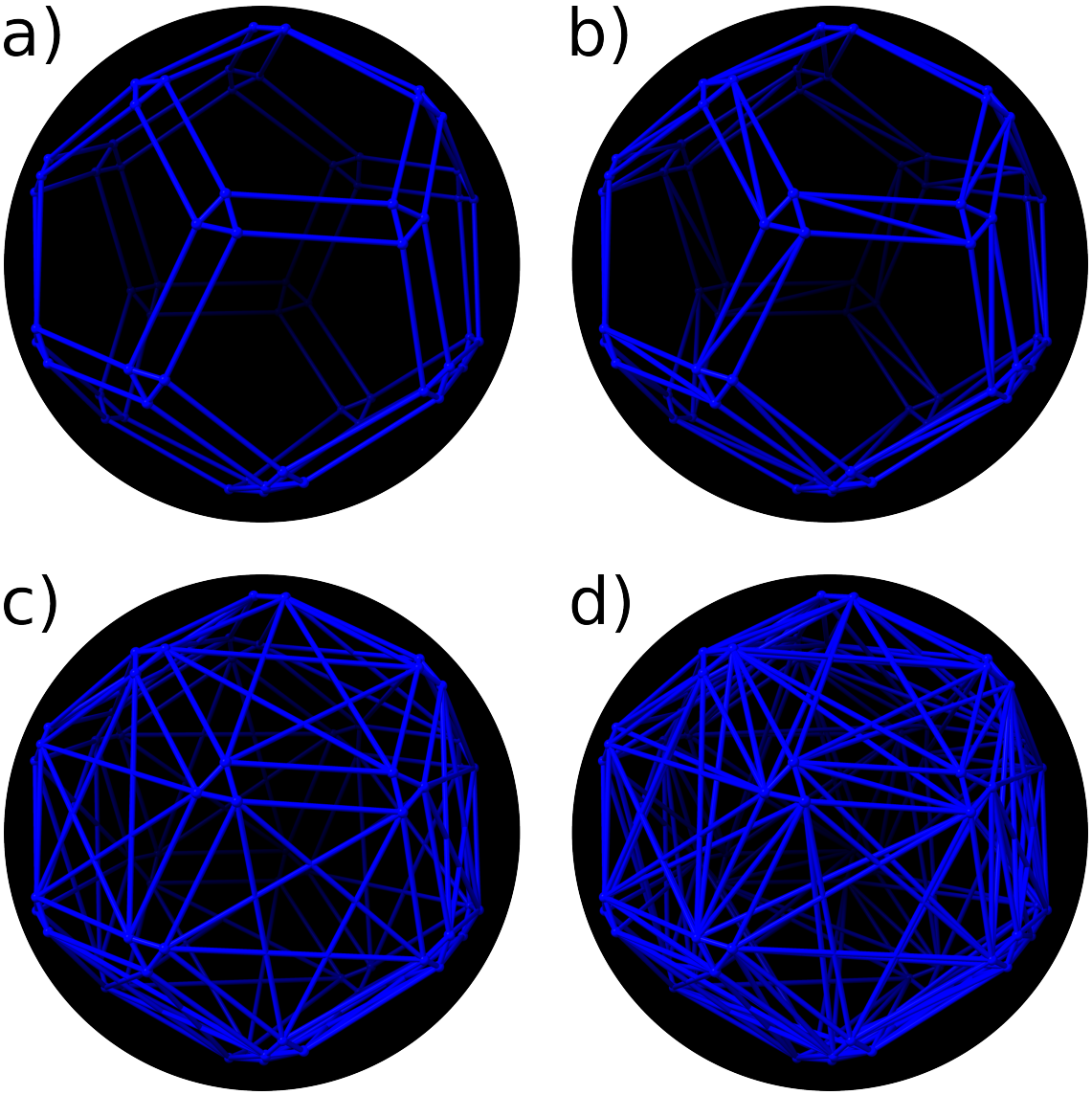

Figure 6.6: A visualisation of four RNA graphs for STNV: (a) {2345}, (b) {12345}, (c) {234589} and (d) {123456789}. Each node represents a binding site and each edge represents the potential paths that an RNA can traverse along.

N	RNA Graph	N_{cap}	N_{path}	Yield
4	{2345}	760	64	38.0 %
4	{2567}	0	1342	0.0 %
4	{2589}	0	12582	0.0 %
4	{3467}	<10	826	<0.5 %
4	{6789}	0	7242	0.0 %
5	{12345}	<10	120086	<0.5 %
5	{12567}	40	12937410	2.0 %
5	{12589}	20	26672730	1.0 %
5	{13467}	80	2240200	4.0 %
5	{13489}	<10	3510708	<0.5 %
5	{16789}	30	12965858	1.5 %
6	{234567}	160	2959693816	8.0 %
6	{234589}	640	2117037620	32.0 %
6	{256789}	0	4399719160	0.0 %
6	{346789}	10	2501657142	0.5 %
7	{1234567}	260	279417463096	13.0 %
7	{1234589}	110	362090241596	5.5 %
7	{1256789}	180	856244868982	9.0 %
7	{1346789}	360	524658774154	18.0 %
8	{23456789}	300	*	15.0 %
9	{123456789}	730	*	36.5 %

Table 6.2: Table containing all potential RNA graphs for STNV's capsid, alongside the degree of all vertices, N, of each node in the RNA graph. The Table contains the initial yield, N_{cap} from a stochastic simulation of the assembly of 2000 copies of an RNA with 30 PSs that had $\Delta G_{rna} = 9$ kcal M^{-1} for each PS. Assembly yield was rounded to the nearest 10, with small but non-zero yields being labelled as < 10. It also contains the number of unique pseudo-Hamiltonian paths that the RNA can trace out on the interior of the capsid, N_{path}, generated using the method described in Section 5.1. The Yield column stores the values of N_{cap} as a percentage of potentially assembled capsids. $*$ indicates that the values for N_{path} were not found for the two RNA graphs with the largest degree, due to the large computational complexity to calculate these values.

of 60 CPs are visited due to mutual exclusion. Thus, RNA traces what will be called a pseudo-Hamiltonian path, whereby the RNA traces out a path where it visits one of the nodes from each mutually exclusive pair exactly once and never visits the other node from the pair. Thus, our depth first search algorithm was adapted to also count these pseudo-Hamiltonian paths and the results are given in Table 6.2 as N_{path}.

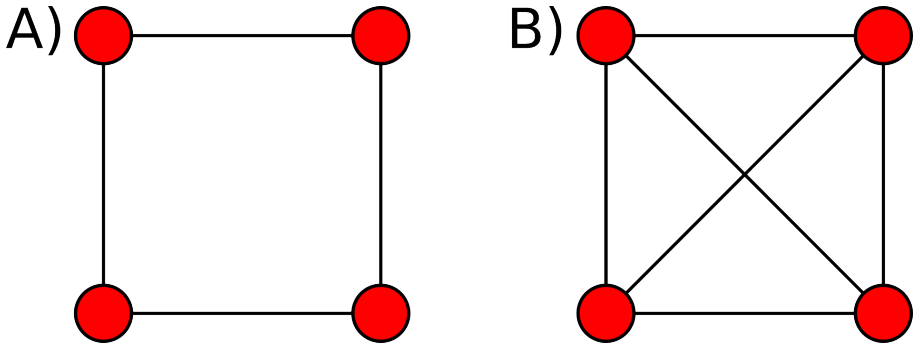

Figure 6.7: This figure features two graphs, where (A) is a spanning subgraph of (B). Consider the Hamiltonian paths (*i.e.* paths where each vertex is visited precisely once) of (A) and (B). As (A) is a spanning subgraph of (B), all paths available to (A) are also available to (B), so (B) will have at least as many Hamiltonian paths as (A), as some will overlap. This is the same for pseudo-Hamiltonian paths on the RNA graphs of STNV.

This table includes all pseudo-Hamiltonian paths that can be generated. Consider the concept of a spanning subgraph, which is a subgraph that contains all of the nodes present in its parent subgraph. As illustrated by Figure 6.7, if there are two graphs (*e.g.* {2345} and {12345}) and one is a spanning subgraph of the other, then all paths available to the spanning subgraph will also be available to the parent graph, thus $N_{path}^{graph} \geq N_{path}^{subgraph}$.

Some parts of this table are intuitive: the RNA graphs with more edges also have more pseudo-Hamiltonian paths. As is shown in Table 6.3, the effect of adding an extra edge to the graph is more impactful when there are fewer edges in total. This shows that the size of pseudo-Hamiltonian paths is large, so if sampling the space using stochastic methods generates the same paths with any regularity, then this suggests that this pseudo-Hamiltonian path's assembly follows a more stable pathway than others. Even small changes to the RNA graph can have very large effects on the potential assembly of STNV, for example the yields of {2345} and {12345}, where adding edge 1 lowers the yield, despite the number of pseudo-Hamiltonian paths increasing.

There are a couple of interesting cases, for example the introduction of edge 1 to RNA graph {2567} gives a much greater effect than adding it to other RNA graphs with the same degree. Also, RNA graph {2345} has far fewer paths than the other RNA graphs with the same degree.

Even Vertex Degree		Odd Vertex Degree		Ratio of $\frac{N_{path}^{Odd}}{N_{path}^{Even}}$
RNA graph	N_{path}	RNA graph	N_{path}	
{2345}	64	{12345}	120086	1876
{2567}	1342	{12567}	12937410	9640
{2589}	12582	{12589}	26672730	2120
{3467}	826	{13467}	2240200	2712
{6789}	7242	{16789}	12965858	1790
{234567}	2959693816	{1234567}	279417463096	94
{234589}	2117037620	{1234589}	362090241596	171
{256789}	4399719160	{1256789}	856244868982	195
{346789}	2501657142	{1346789}	524658774154	210

Table 6.3: A table illustrating the effect that adding a single extra edge (edge 1) to RNA graphs has on the number of pseudo-Hamiltonian paths. The ratio of N_{path} for the RNA graph that includes edge 1 over the RNA graph that does not include it is given to the nearest whole number. Typically, adding extra edges is more impactful for RNA graphs with smaller degrees. The order of magnitude of this ratio is mostly consistent within a given vertex degree. The one exceptional case is with {2567}, where the effect of the addition of edge 1 to obtain {12567} is significantly higher than for other graphs with the same degree.

6.5 OPTIMISATION OF STNV ASSEMBLY EFFICIENCY

Initially, there is a collection of 21 different RNA graphs that could feasibly represent the assembly mechanism that STNV uses to assemble. Some are likely to be more efficient than others at generating completed capsids, so stochastic simulations were used to model the assembly. Initially, this was a simulation of 2000 uniform RNAs, where $\Delta G_{rna}(n) = 9$ kcal M^{-1} for all n. Here, $\Delta G_{rna}(n)$ represents the interaction free energy between the n^{th} PS on the RNA and a CP. The number of capsids formed during this simulations is shown as N_{cap} in Table 6.2 and the percentage yield is also shown in the final column. Some RNA graphs are more effective than others and some did not result in any completed capsids at the end of the simulation.

The three most effective RNA graphs are: {2345}, {234589} and {123456789}, generating a yield of $> 30\%$. This is still a lower yield than is to be expected for a virus, so one might expect that there are parameters in the simulation that can be altered to increase the yield of this further. As has been shown by Dykeman *et al* [19], RNAs with uniform $\Delta G_{rna}(n)$ values tend not to give the largest yields in stochastic simulations of viruses. Thus, altering the values of $\Delta G_{rna}(n)$ is one possible way to optimise the number of capsids assembled. This optimisation is a

time consuming process, so it is not feasible to optimise all of the different RNA graphs and therefore the three RNA graphs that generated the largest yields ({2345}, {234589} and {123456789}) were chosen. Additionally, {12345} was also chosen, in order to observe the effect of adding 1 edge to {2345}, as adding an edge would not be expected to have such a detrimental effect on the initial yield as was observed in Table 6.2. Lastly, due to it having an above average yield and also not containing edges 2 and 5, that each of the others did, {1346789} was also chosen, to see how the removal of this pair of edges affected the optimisation.

In order to optimise the PS distribution (*i.e.* the distribution of $\Delta G_{rna}(n)$ values) of RNAs to increase yield, a genetic algorithm was used. Initially, a set of 1024 randomly generated RNAs containing 30 randomly generated $\Delta G_{rna} \in \{4..11\}$ were used. These were produced by generating a random integer in the interval {4..11}, where every integer had an even probability and setting $\Delta G_{rna}(n)$ to this value for each n. Once the PSs have mutated, the range of $\Delta G_{rna}(n)$ is allowed to be {4..12}, so RNA can evolve the strongest PSs, rather than just start with them. This will generate one distinct RNA and the process was continued until a set of 1024 random RNAs was generated. The parameter space of the PSs is 30-dimensional, so it is unlikely that these simulations have sampled the entire space. However, there are some tests that can be done to demonstrate that it sampled a range of different distributions. To do this, all RNA PSs with $\Delta G_{rna} \in \{4..7\}$ are labelled as low PSs and all $\Delta G_{rna} \in \{8..11\}$ are labelled as high PSs. Each PS has its $\Delta G_{rna}(n)$ values stored to the nearest 0.1 kcal M^{-1} and whilst they are initially all integers, once optimisation occurs, they may take on non-integer values. Then, the randomly generated PSs are analysed to see how many of the 1024 RNAs had a low PS in each of its positions. The median value of this was 512 and the values were all in the region close to this, suggesting that each position in the RNA had a fair chance at being both a low and a high PS. Next, the total number of weakly bonding PSs present in each RNA was calculated. These results approximately show a binomial distribution, centred at 15 low PSs, so the space of PS distributions, where many of the PSs are strongly binding or many are weakly binding, is not as well explored as when their numbers are more even. This data is presented in Figure 6.8.

To allow each of the RNA graph to be optimised, a separate simulation was ran for each of the random RNAs, where 2000 copies of the RNA were allowed to assemble in a volume ($0.7\mu m^3$) representative of a small bacterial cell or cellular compartment. In Figure 6.9, the initial yield from the first set of randomly generated

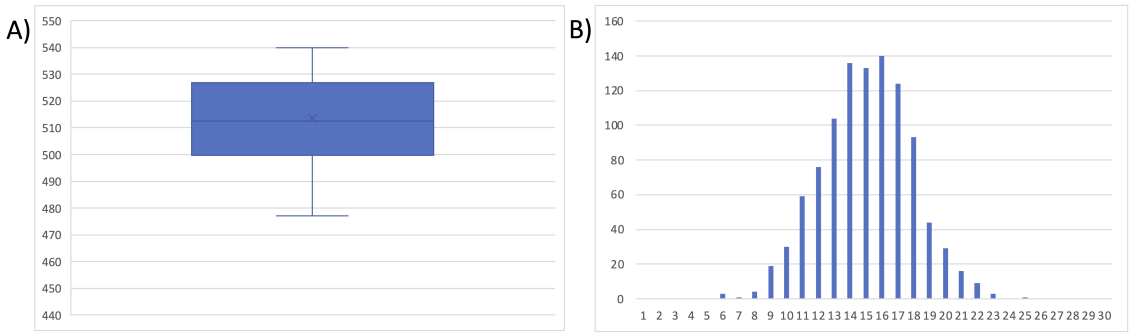

Figure 6.8: Every PS on each of the 1024 randomly generated RNAs was labelled as either weakly binding (if $\Delta G_{rna}(n) < 8.0$) or strongly bonding (if $\Delta G_{rna}(n) \geq 8.0$). This data was analysed to see how well the parameter space of PSs was sampled. (A) The number of times a strong binding PS was observed at each of the 30 positions across the RNA was tallied. This box and whisker chart shows the range of values that this count achieved, with the median value at 512.5. This means that each PS took both large and small values of $\Delta G_{rna}(n)$, suggesting this was sampled well. (B) The number of strongly binding PSs on each RNA was tallied. This bar chart shows the number of RNAs that had a given number of strongly binding PSs. It resembles a binomial distribution, centred about 15 high PSs in an RNA. This means that RNAs with small numbers of strongly binding PSs or large numbers of strongly binding PSs was not well sampled, though RNAs with a mix of strongly binding and weakly binding RNAs was.

RNAs is shown, for four of the choices of RNA graph. For each RNA graph, once the simulations were completed, the 256 RNAs with the top yield were identified and collected. These RNAs were mutated, to generate 768 new RNAs and the simulations were restarted for this new collection of 1024 RNAs. During these mutations, the values of $\Delta G_{rna}(n)$ were always limited to the range of 4 kcal M^{-1} to 12 kcal M^{-1}. Each PS had a 4% chance of changing to any value in the range [4.0,12.0] (to 1 decimal place) and a 40% chance of a mutation, where it would change value by ± 0.1 kcal M^{-1} or ± 0.2 kcal M^{-1}.

This was repeated many times and after 20-30 generations, the top 20 RNAs all had similar yields of completed capsids and no new RNAs were joining the top 20. A cursory inspection of the top RNAs showed that they were all highly similar to each other. For each RNA graph, extra simulations of the top RNA were ran, to generate a more accurate yield of the assembly process for this RNA. The percentage yield is stored in Table 6.4, alongside the yield from other RNAs. The PS distribution of the optimised RNAs for each RNA graph is shown in Figure 6.10.

The one exception to this process was RNA graph {12345}, for which an optimised

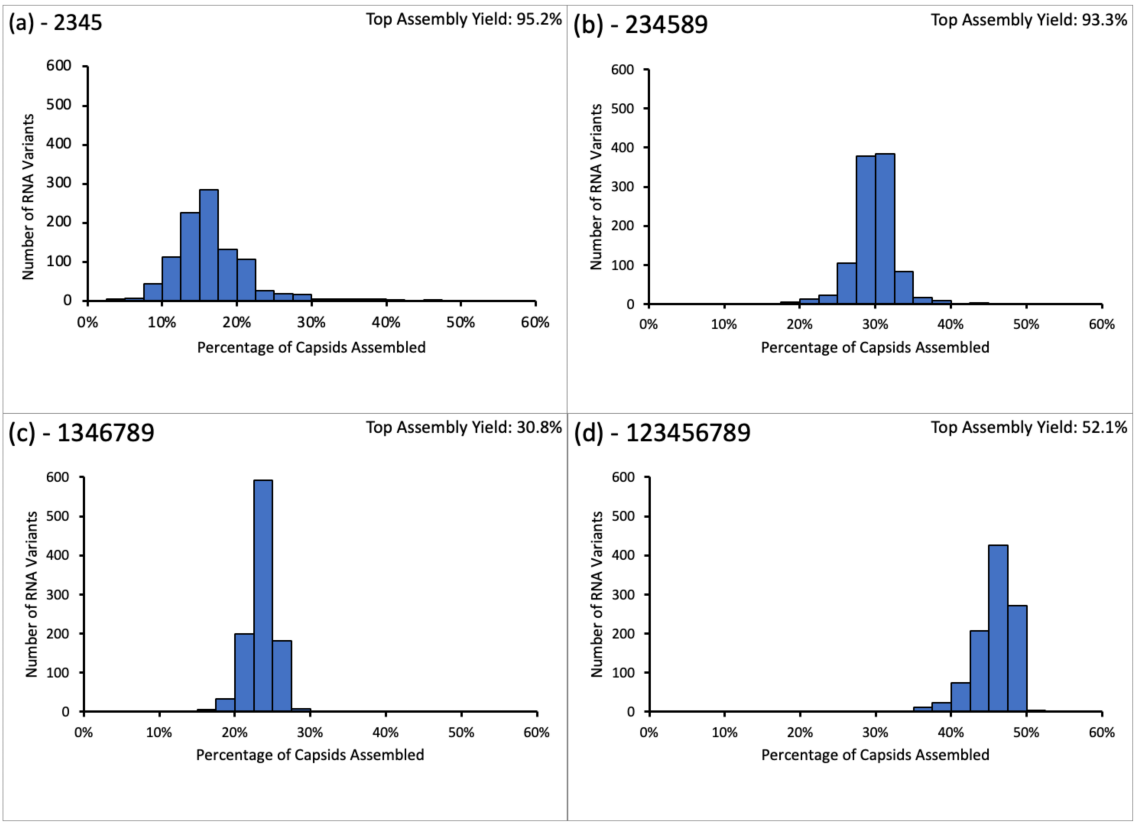

Figure 6.9: A histogram containing the yields from the first set of simulations using the 1024 random RNAs, with RNA graphs (a) {2345}, (b) {234589}, (c) {1346789} and (d) {123456789}. Due to its highest yield being $< 1\%$, the histogram for {12345} was not produced, though optimisation did improve upon this. Each simulation contained 2000 copies of its RNA and enough proteins to make 2000 completed capsids. None of these initial RNAs got more than 60% of the potential capsids produced. This shows that usually there was a spread in the yields for these capsids, dependent on the RNA and there was always a tail of both higher and lower efficiency RNAs. The post-optimisation yield for each RNA graph is shown in the top right of each histogram.

RNA Graph	4	8	10	Optimised	Knockout	Nuc Only
{2345}	30.72%	35.68%	34.93%	95.20%	3.88%	93.43%
{12345}	2.28%	0.08%	0.15%	6.69%	0.26%	4.23%
{234589}	22.14%	32.88%	31.62%	93.33%	9.62%	65.73%
{1346789}	12.00%	17.57%	17.86%	30.77%	21.26%	14.38%
{123456789}	28.24%	34.04%	35.17%	52.06%	37.86%	30.62%

Table 6.4: Table containing the assembly yields for the five optimised RNA graphs, using a variety of RNAs. The first three are uniform PS distributions, where $\Delta G_{rna}(n) = 4, 8, 10$ kcal M^{-1} for all n, respectively. Next is the yield from the optimised RNA generated using the genetic algorithm described in Section 6.5. For the final two RNAs, more explanation is needed. Each optimised RNA (shown in Figure 6.10) had a series of strongly binding sites adjacent to each other, believed to represent the sites where the capsid nucleates from. Knockout is used to model the removal of the nucleation site, which is done by setting $\Delta G_{rna}(n) = 4$ kcal M^{-1} for all PSs in the series of strongly binding PSs. Lastly, there is Nuc Only, which represents the RNA where the optimised RNAs' nucleation site is preserved but all other $\Delta G_{rna}(n)$ values are set to 4 kcal M^{-1}. This tests the impact of having the nucleation site present, when compared to the other optimised PS binding strengths. As an example, all of the RNAs used for {2345} are shown in Figure 6.11.

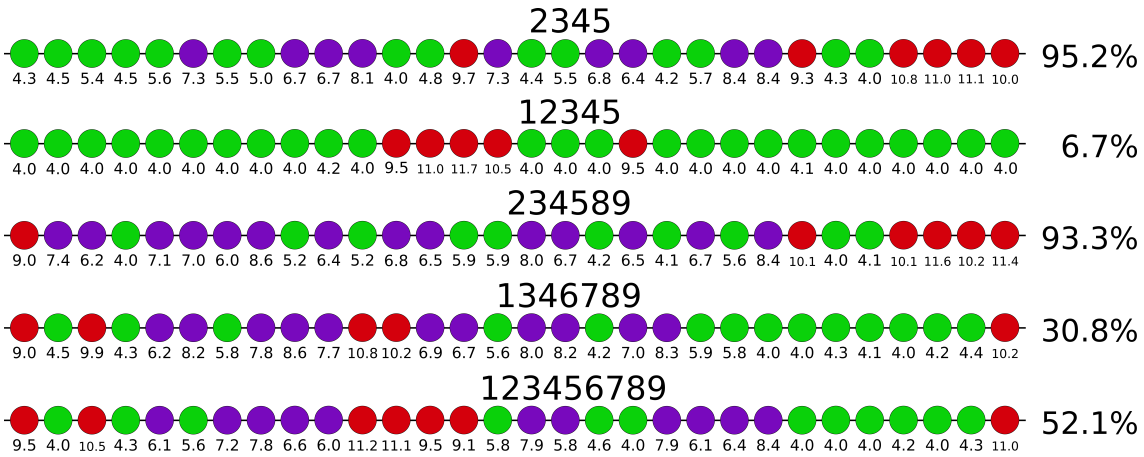

Figure 6.10: Once the optimisation process was completed, the best RNAs for each RNA graph were gathered. The PSs are shown here as a series of beads, where red represents a strongly binding PS (9.0 kcal M^{-1} $\leq \Delta G_{rna}(n) \leq$ 12.0 kcal M^{-1}), purple PSs bind with a medium strength (6.0 kcal M^{-1} $\leq \Delta G_{rna}(n) <$ 9.0 kcal M^{-1}) and green PSs are weakly binding (4.0 kcal M^{-1} $\leq \Delta G_{rna}(n) <$ 6.0 kcal M^{-1}). The values of $\Delta G_{rna}(n)$ are shown beneath each bead. It also features the yield, generated from the average yield across five simulations of 2000 RNAs and sufficient proteins to fully assemble a capsid on each RNA.

RNA with low yield was generated via the method above. During later analysis, a better PS distribution was generated by reducing all but 4 strongly bonding PSs' $\Delta G_{rna}(n)$ to 4 kcal M^{-1}. Thus, the genetic algorithm was applied to this RNA to see if it could be further improved, which led to a small increase in yield and the RNA presented here. This does suggest that there is a large space of PS distributions that has not been sampled, so there may be better PS distributions out there. It could also be a result of the low initial yield of {12345}, as the genetic algorithm is more susceptible to small fluctuations when the top yield from a random RNA was less than 20 capsids out of 2000.

6.6 ANALYSIS OF OPTIMISED RNA RESULTS

6.6.1 EFFECT OF CHANGING THE PS DISTRIBUTION

As can be seen in Table 6.4, the end result of the optimisation has a significant difference on yield depending on the RNA graph. The top two RNA graphs are {2345} and {234589}, which both generated a yield above 90%.

Looking at the uniform RNA's yield, most RNA graphs have their lowest yield in Uniform 4, as shown in Figure 6.11. The only exception is {12345}, which has its best yield at this value. This could suggest that this RNA graph is prone to getting stuck in kinetic traps. This is believable as it does have a very low yield, compared to other RNA graphs. Uniform 8 and 10 are both very similar in efficiency, differing by approximately 1 percentage point across all RNA graphs.

Considering the PS distributions shown in Figure 6.10, it is noticeable that each of the optimised RNAs have a small section of 2-4 consecutive strongly binding PSs. Strongly binding PSs are usually the part of the RNA that the capsid nucleates from [19, 43]. To examine if this is always the case, knockout simulations were performed. These simulations take inspiration from work by Patel *et al* [42]. In this work, they showed that by reducing the binding strength of the PSs where the assembly usually begins (*i.e.* the nucleation site), the assembly efficiency of the capsid was significantly reduced. Thus, the optimised RNAs for each RNA graph were modified, setting all of the 2-4 strongly binding PSs to be weakly binding. An example of how {2345}'s RNA changes is shown in Figure 6.11. The assembly of these knockout RNAs was simulated and the results were shown in Table 6.4. The results show that the changes to these PSs significantly impacted the yield for RNA

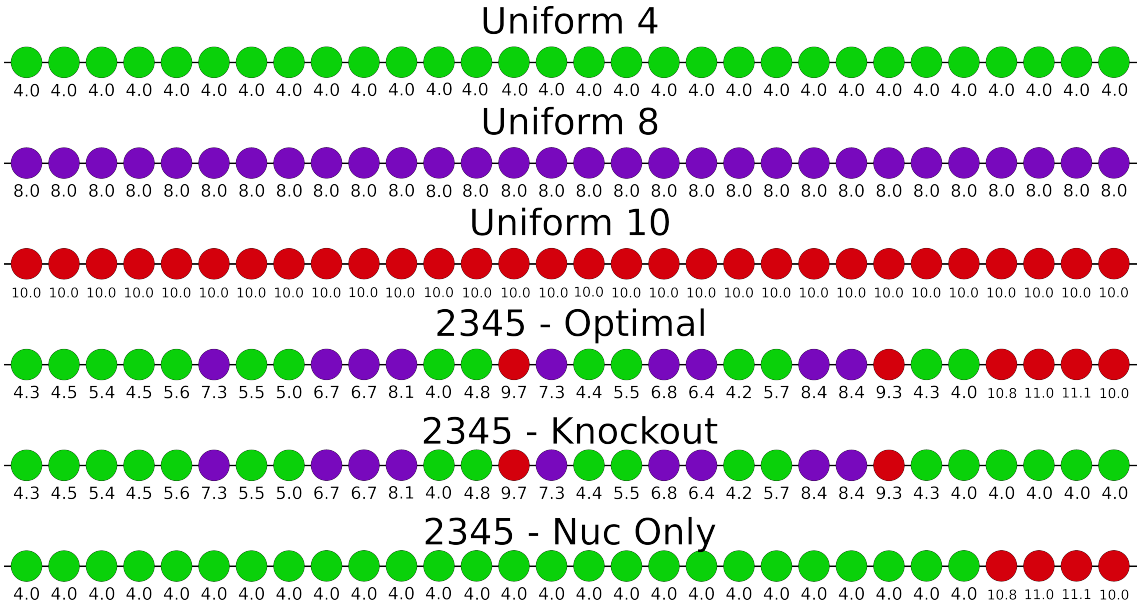

Figure 6.11: Table 6.4 shows the yield for simulations involving a variety of RNA graphs using a few different RNAs. This figure contains all of the RNAs used for RNA graph {2345}, including the uniform RNAs, that were common across all RNA graphs. It also features the optimised RNA, that was produced using a genetic algorithm (and is unique to each RNA graph) and two variants: Knockout and Nuc Only. Knockout is generated by taking the cluster of 2-4 adjacent strongly binding PSs that features in all optimised RNAs and reducing their $\Delta G_{rna}(n)$ to 4 kcal M^{-1}. Nuc Only is generated by taking the optimised RNA and changing the $\Delta G_{rna}(n)$ of all PSs except the strongly binding cluster to 4 kcal M^{-1}.

graphs {2345}, {12345} and {234589}, but less so for the other RNA graphs. The three that were most affected were the three with the most clearly defined nucleation site, as {1346789} and {123456789} both had a cluster of mid-strength PSs near the removed nucleation site. Considering the Nuc Only RNA, the opposite effect is observed. Nuc Only RNA is where the optimised RNA has had all of the binding sites other than the 2-4 strongly binding PSs set to be weakly binding (an example is shown for {2345} in Figure 6.11). RNA graphs {2345}, {12345} and {234589} are impacted much less than in knockout, with {2345} still getting over 90% yield. Both {1346789} and {123456789} had a larger decrease to their yields than the others, suggesting that the mid-strength PSs are important to their assembly.

Additionally, comparisons can be applied to the PS distributions more closely. Comparing the right half of {2345} and {234589}, both bear a strong resemblance to each other, with only 2 of the rightmost 12 PSs not sitting within the same

weak/mid/strong strength bond grouping. These two RNA graphs are similar to each other in terms of which edges are present in the graph, as only two extra edges (8 and 9) are added to {2345} to make {234589}. Also, the optimised yield of both RNA graphs is similar. Note that, {1346789} and {123456789} also bear similarities to each other, with only 8 of the 30 PSs not sitting within the same weak/mid/strong strength bond grouping, as defined in Figure 6.10.

Looking at Figure 6.9, the spread of yields for the initial, randomly generated RNAs is shown. For RNA graphs {1346789} and {123456789}, the optimised yield is not much greater than the top initial yield, which is not the case for {2345} and {234589}. This suggests that despite the optimised yield being higher than the uniform yields, the optimisation did not manage to generate the same level of improvement for RNA graphs {1346789} and {123456789} as it did for the others. Their histograms had much smaller tails than those of {2345} and {234589}. These larger tails meant that the sampled set of RNAs contained a small number of RNAs that were effective at assembling either {2345} and {234589}, which allowed the genetic algorithm to take the effective RNAs and improve on them until they were efficient at assembling.

Overall, this suggests that RNA graphs with greater vertex degrees have more flexibility, so are more able to assemble regardless of the RNA but that a smaller vertex degree assists with generating the highest yields, potentially due to the more confined path choices.

6.6.2 Observed Frequency Of Sampled Paths

To assess if there is a preferred RNA organisation and assembly pathway, further tests were done. One form of analysis was looking at the paths traced out on the RNA graph, which indicate the layout of RNA on the interior of the capsid. This analysis was completed using the techniques described in Section 5.2.1, to compare and see if any paths are more common in the output from the optimised RNA's yield on each RNA graph. The results were interesting and can be seen in Figure 6.12 , where some RNA graphs have a strong bias towards a small subset of the potential paths. This was most noticeable in RNA graph {2345} but was also present in both {12345} and {234589}. Interestingly, the similar paths assembled by {2345} and {234589} were highly similar to each other, with only 4 of the 29 edges in the path differing. These were the edges that traversed each point about a 5-fold axis, where

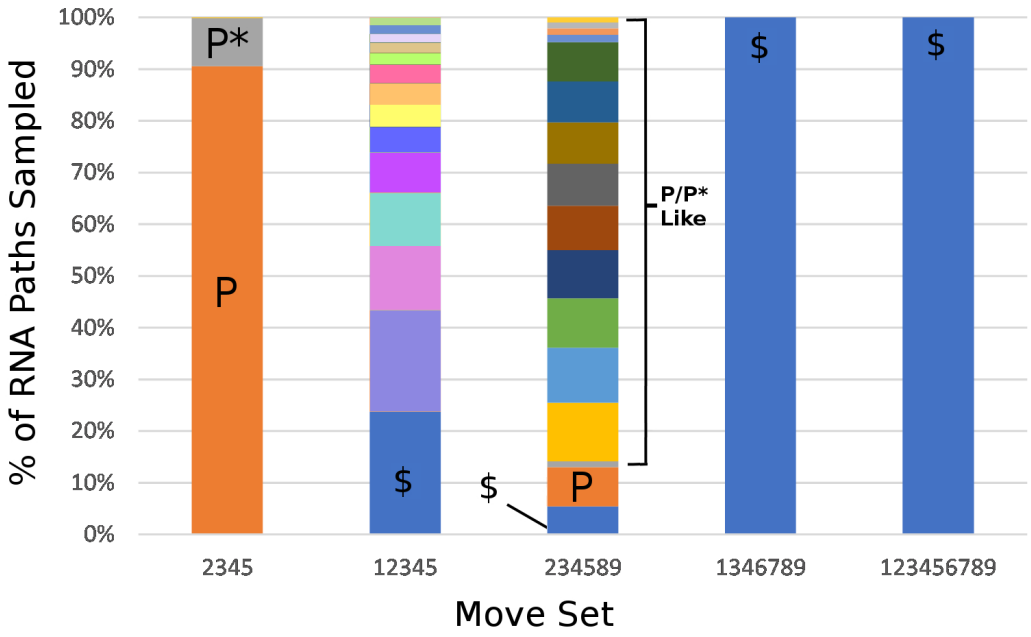

Figure 6.12: For each RNA graph, five simulations including 2000 copies of its optimised RNAs was run. The results were analysed to test for any repetition in the paths the RNA traced out on the interior of assembled capsids. This lead to a sample of 9520 paths for {2345}, 669 paths for {12345}, 9333 paths for {234589}, 3077 paths for {1346789} and 5206 paths for {123456789}. The chart shows the frequency certain paths were assembled during these simulations. Any path that occurred 10 or fewer times has been clustered into the box labelled by $. RNA graph {2345} showed a strong frequency of two highly similar paths, labelled as P and P*, out of the space of 64 different paths it could have assembled with. P and P* only differ by whether they went clockwise or anticlockwise about the first 5-fold axis in the path, the rest of the path is identical. RNA graph {234589} also showed a high frequency of paths that were similar to P and P*, with additional initial paths about the initial 5-fold axis but outside of the initial 5-fold, the paths all matched P. This is within a much larger space of 2117037620 potential paths, so the correlation is significant. In its small number of paths, {12345} also shows a noteworthy amount of similarity between the paths it sampled, of 120086 potential paths, though these similar paths were different to those from {2345}. This figure is reproduced from Hill *et al* [32].

it is thought to have nucleated. After this, the path generated by the assembly was identical. The most frequently occurring version of this path (labelled P in Figure 6.12) is shown in Figure 6.13.

That this path is found exclusively in the two RNA graphs, that are also the graphs that have the highest yield is unlikely to be coincidence, due to the large space

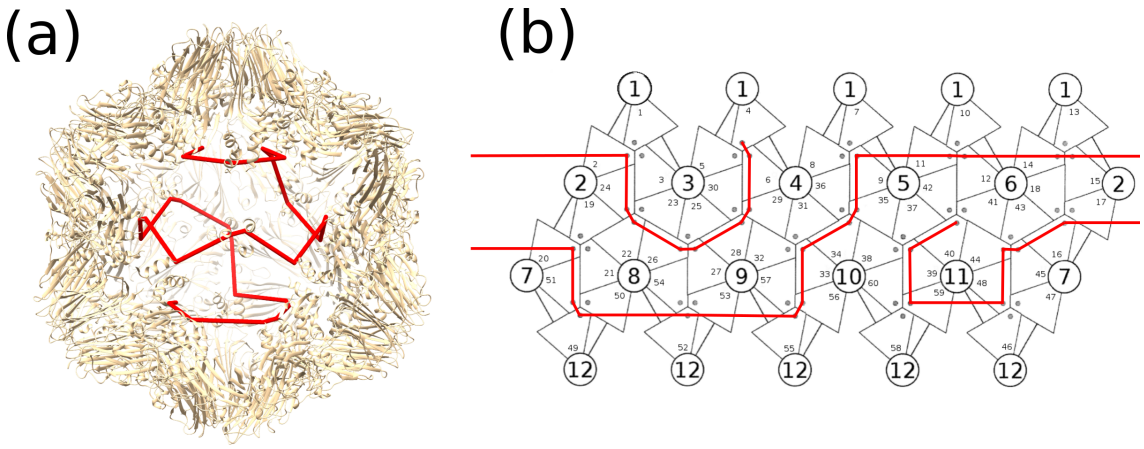

Figure 6.13: During assembly simulations of RNA graphs {2345} and {234589}, the RNA reliably traced out paths that were highly similar. These similar paths consisted of over 99% of the paths assembled for {2345} and over 90% of those for {234589}. This figure shows the most frequently assembled path, labelled as P in Figure 6.12. The only difference between P, P* (the second most frequent) and similar paths was how it traversed the initial 5-fold axis, which it nucleated around. (a) The path traced out on the inside of an STNV capsid corresponds to the shortest path between CP binding sites and this is an idealised mathematical approximation of the more complex molecule in nature. The initial 5-fold axis is shown at the top. (b) The path traced out on a net of the external surface of the capsid. The initial 5-fold is labelled as 11 on this net.

of paths identified combinatorically. This suggests that this path's intermediates are more stable than for other available paths and that this path has a resistance to kinetic traps which can stifle assembly. Interestingly, this path connects opposite 5-fold vertices in a spiral winding around the inside of the capsid (Figure 6.13.

The paths from the knockout simulations for {2345} were analysed, to contrast to the optimised RNA's simulated paths. These showed a strong degree of similarity in the generated paths, with 388 capsids assembled using only 6 paths. One path made up 78.9% of the assembled paths and two other paths (which were highly similar to each other) made up 10.3% and 8.8%. These paths seem to have nucleated in the middle of the RNA, rather than at one end, as the optimised RNA did. None of these paths matched the path that the optimised RNA preferentially chose during assembly, so not only did knocking out the strong PSs dramatically reduce the yield, it also fundamentally changed the assembly of the capsid.

Manual analysis of the path within {2345} was performed, starting with the first pentamer formed about a 5-fold axis. It was assumed that after the RNA visits a

node, as soon as that node's mutually exclusive partner can join the structure, it will immediately do so. The analysis then checked, step by step, what deviations from P were able to form. This analysis showed that, once the first pentamer was formed, then there was no choices for the RNA to assemble into any other arrangement than P. There are a few reasons that lead to this. The first is that the RNA graph only has 4 edges per node and that to form a path, it needs to enter and leave each node, so there are at most 3 choices at each node. Next, the mutual exclusion removes a large number of potential nodes that could be visited, so there are fewer choices in total. Lastly, the factor that was not considered in the initial combinatoric analysis is that the RNA can only visit a node (add a capsomer bound to a PS to the structure) if that node corresponds to a capsomer adjacent to the existing intermediate structure. Along the assembly of P, there are only 6 possible deviations from P that are allowed on the RNA graph and none of these would be consistent with the CP positions occupied in the intermediate structure as encoded by the capsomer graph.

Similar analysis of the path within {234589} was also performed and it gave similar results. Once the pentamer was formed, there were 6 potential deviations from P that were allowed by both the RNA graph and the capsomer graph. Each of these relied upon at least one protein that was not bound to the RNA joining the structure (and only forming one bond) and staying in the structure long enough that a PS-bound protein could attach to it. During the simulations unbound proteins with only one neighbouring protein tended to be quite short lived, so these deviations are possible but thermodynamically unfavourable, hence were not observed regularly.

This analysis suggests that with the right start, it is highly likely that these two RNA graphs will assemble into capsids where this path is traced out. It also shows why the knockout simulations showed so much more of an impact to these two RNA graphs than Nuc Only: all the RNA needs to do to begin assembly of capsids with this path is form a pentamer and then there are a very limited number of kinetic traps that need to be avoided. Other paths were observed in both RNA graph's assembly, so it is not the only path that could occur, just the most effective path to assemble along.

Note also that after the RNA graph has a degree of 7 or more, despite the optimisation, no paths appear to be repeated, most likely due to the complexity generated by the number of choices of edge at each node. By contrast, the RNA graphs with smaller vertex degrees all show path preference to some extent. Whilst {2345} and {234589} are both efficient assemblers, {12345} is not, so the repeated

paths cannot be attributed solely to the efficiency, suggesting that the degree of the vertices is a contributing factor as well. This could be interpreted as RNA graphs with too large a space of paths are less able to develop a PS distribution that can guide the assembly down one specific path and so cannot achieve the same efficiency that RNA graphs with smaller degrees can achieve.

Overall, the simulations that generate the highest assembly yield and the simulations which align closest to experimental results, are those with either RNA graph $\{2345\}$ or $\{234589\}$. From this, it may be inferred that these RNA graphs represent the way that the RNA is organised on the inside of STNV's capsid during assembly. This is a conclusion that would be difficult to determine using experimental work alone, highlighting the importance of mathematical modelling in this area.

— 7 —

A General Model Of $T = 3$ Capsids

The previous chapter covered the assembly of STNV, a $T = 1$ virus. Following from this, it makes sense to try to apply those same novel techniques to a larger virus. After $T = 1$, the next largest viral capsids within Caspar-Klug classification are $T = 3$ viruses. There are a large number of $T = 3$ viruses, including STNV's helper virus TNV (as a satellite virus, STNV cannot assemble unless it co-infects a cell alongside its helper virus). So studying $T = 3$ viruses is the logical next step in developing our model of assembly.

This means that each capsid will be made up from 180 copies of the coat protein. According to Viral Tiling Theory (covered in Section 2.3), there exist three different tilings that tile the $T = 3$ capsid, which are the Kite, Tri and Rhombic tilings. This is due to the capsids being made of 180 CPs that obey icosahedral symmetry via quasi-equivalence [14]. These tilings can be seen in Figure 7.1 on planar nets. For the Kite and Tri tilings, we assume that the capsomers the capsid assembles from are kite and triangular trimers (*i.e.* capsomers made of three CPs). For the Rhombic tiling, we assume these capsomers are rhombic dimers (*i.e.* capsomers made of two CPs). Each of these tilings differ from each other in the number of capsomer-capsomer interactions per capsomer, the number of capsomers per capsid and also the total number of capsomer-capsomer interactions per capsid, which are shown in Table 7.1. These differences in the tiling's geometry lead to differences during the assembly of viruses with these geometries. During this chapter, many different simulations will explore how these differences in geometry affect the assembly. As a result, this chapter will consider hypothetical viruses that fit these tilings, rather than any specific virus. Examples of viruses that fit the three tilings are: MS2 uses the Rhomb tiling, Tobacco Ringspot Virus uses the Kite tiling and Pariacoto Virus used the Tri

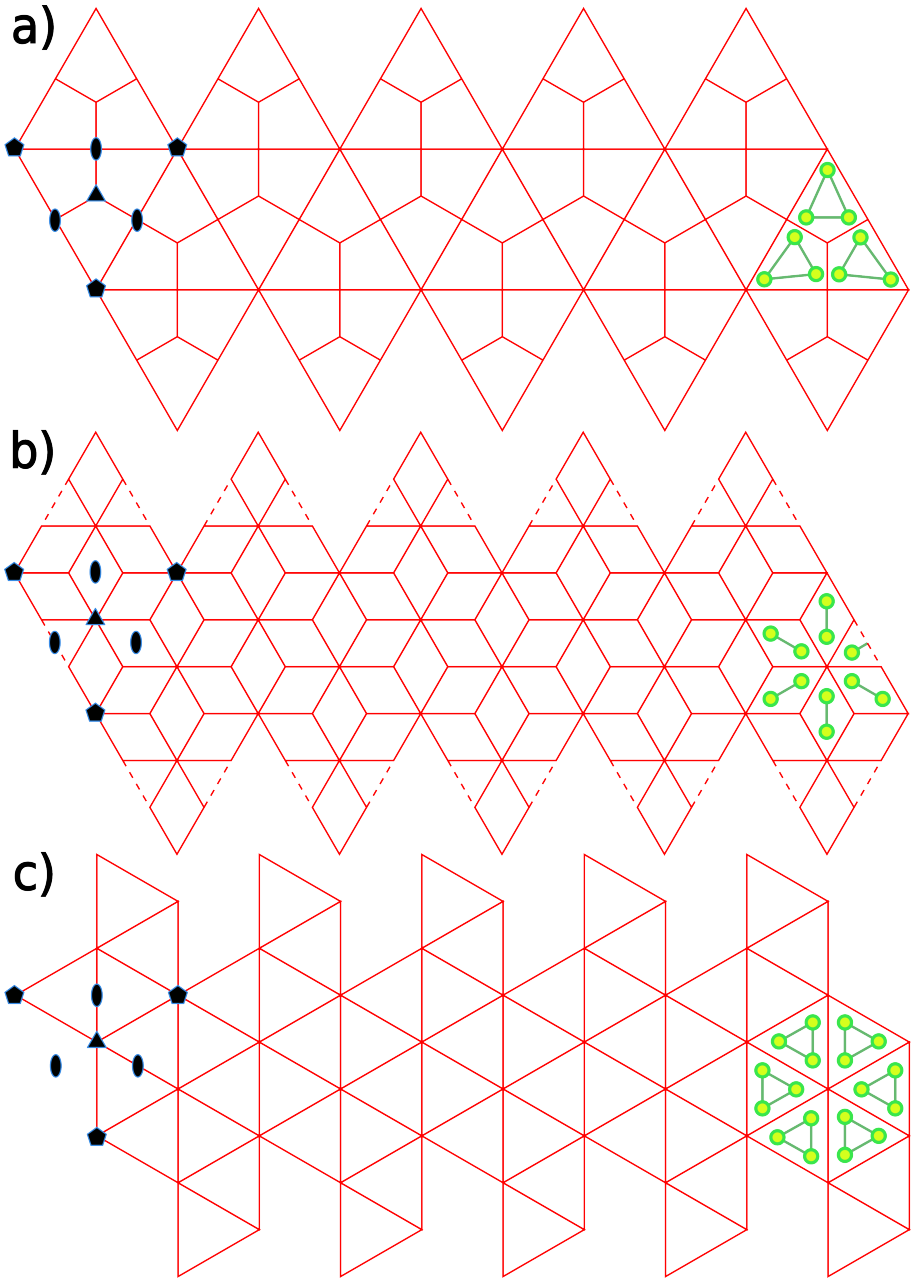

Figure 7.1: Three planar icosahedra that all show a different tiling of the capsid. The tilings are called the (a) Kite, (b) Rhomb and (c) Tri tilings. On the left of each tiling, the symmetry axes are shown as a pentagon, triangle and oval for 5-fold, 3-fold and 2-fold symmetry axes, respectively. On the right hand side are approximate positions of where proteins in this tiling might sit (green).

Tiling	Interactions/Capsomer	Capsomers/Capsid	Interactions/Capsid
Tri	3	60	180
Kite	4	60	240
Rhombus	4	90	360

Table 7.1: Table listing the number of capsomer-capsomer interactions in a complete capsid for each $T = 3$ tiling.

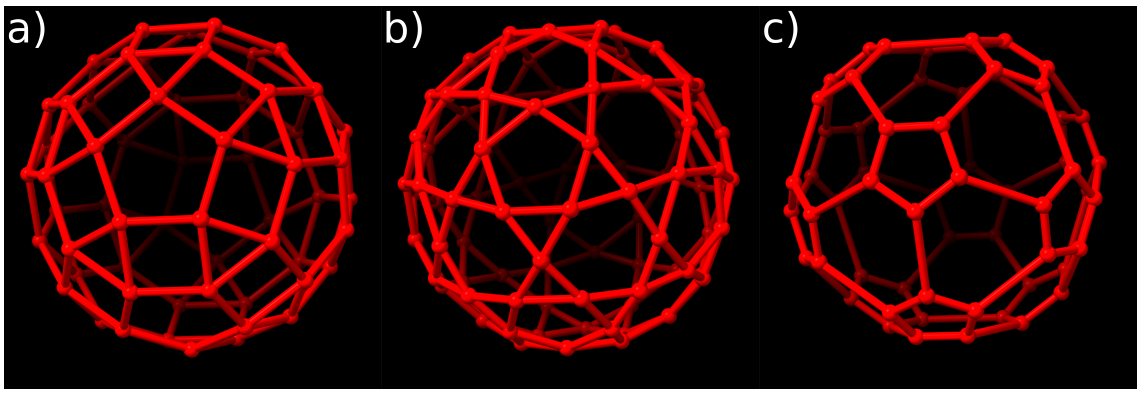

Figure 7.2: The capsomer graphs are shown, for each of the $T = 3$ tilings. These tilings are shown in planar form in Figure 7.1. These tilings are named the (a) Kite, (b) Rhomb and (c) Tri tilings. To produce these capsomer graphs, each capsomer (*i.e.* each trimer in the Kite or Tri tilings and each dimer in the Rhomb tiling) is assigned a node in the space, placed roughly at the centre of where the capsomer would sit in 3D. Then, every capsomer-capsomer contact is encoded as an edge between these nodes.

tiling. There are other examples of viruses that use these tilings but none of the virus-specific aspects of assembly (such as STNV's mutual exclusion, used in the previous chapter) from any of them will be applied to this model in this chapter.

Another important change that will also be covered in this chapter is the differences between assembly simulations in the presence and absence of RNA. RNA binds to capsomers and can help stabilise the structure of intermediate capsids but it can also restrict the ways in which the capsid can assemble. So analysis of both the presence and absence can give different insights into viral assembly.

7.1 ASSEMBLY IN THE ABSENCE OF RNA

Initially, work on the assembly of $T = 3$ viruses was in the absence of RNA. This meant that any results from the simulations would be exclusively due to the geometry

of each capsomer and the associated adjacencies, contained within its capsomer graph. Due to this, a value of ΔG_{bond} must be found for each tiling, to ensure that they are all on a level playing field when the RNA is present. Figure 7.2 shows the capsomer graphs, generated from the three $T = 3$ tilings, as shown in Figure 7.1. Results for the capsomer graphs will be used as a baseline for later work in the presence of RNA.

7.1.1 Comparisons Between Different Tilings

In the initial simulations, it was assumed that the ΔG_{bond} value of each capsomer-capsomer interaction was equivalent to all others. This means the relative stability of an intermediate can be found just by counting the total number of capsomer-capsomer interactions. In the absence of RNA, ΔG_{bond} can easily be optimised to find the value that gives the best yield for a given tiling.

The simulations investigated two different assembly scenarios: the first was where the full amount of capsomers were present at the beginning of the simulation and the second being where the capsomers are produced gradually as the simulation ran. These will be referred to as simulations without a ramp and simulations with a ramp, as the total number of capsomers present in a simulation is ramped up when the capsomers are produced gradually. These can give insight into *in vitro* and *in vivo* processes respectively, as earlier *in vitro* experiments start with proteins being mixed together into a solution whereas *in vivo* experiments have the cells gradually produce capsomers over a longer period of time, a scenario that is now also mimicked by many *in vitro* experiments. The rate at which capsomers were produced was initially $400s^{-1}$ but when the rate was reduced to $200s^{-1}$, the yield did not noticeably change, suggesting that the details of the ramp are not important for the outcome. In both simulations, there were enough capsomers present in the solution at the end of the simulation that it could have formed up to 2000 completed capsids.

The results of these simulations can be seen in Figure 7.3 for Kite, Rhomb and Tri tilings. The percentage yield after the simulation was assumed to have reached equilibrium, is shown against the strength of the capsomer-capsomer interactions (ΔG_{bond}) in the structure for both ramped and non ramped simulations. It is worth noting that, for lower ΔG_{bond} values, the two simulations give very similar results. However, if ΔG_{bond} continues to increase, then the simulations with a ramp will plateau and the simulations without one will have their yield sharply decrease. This is due to the capsomers being more easily able to nucleate, thus forming more than

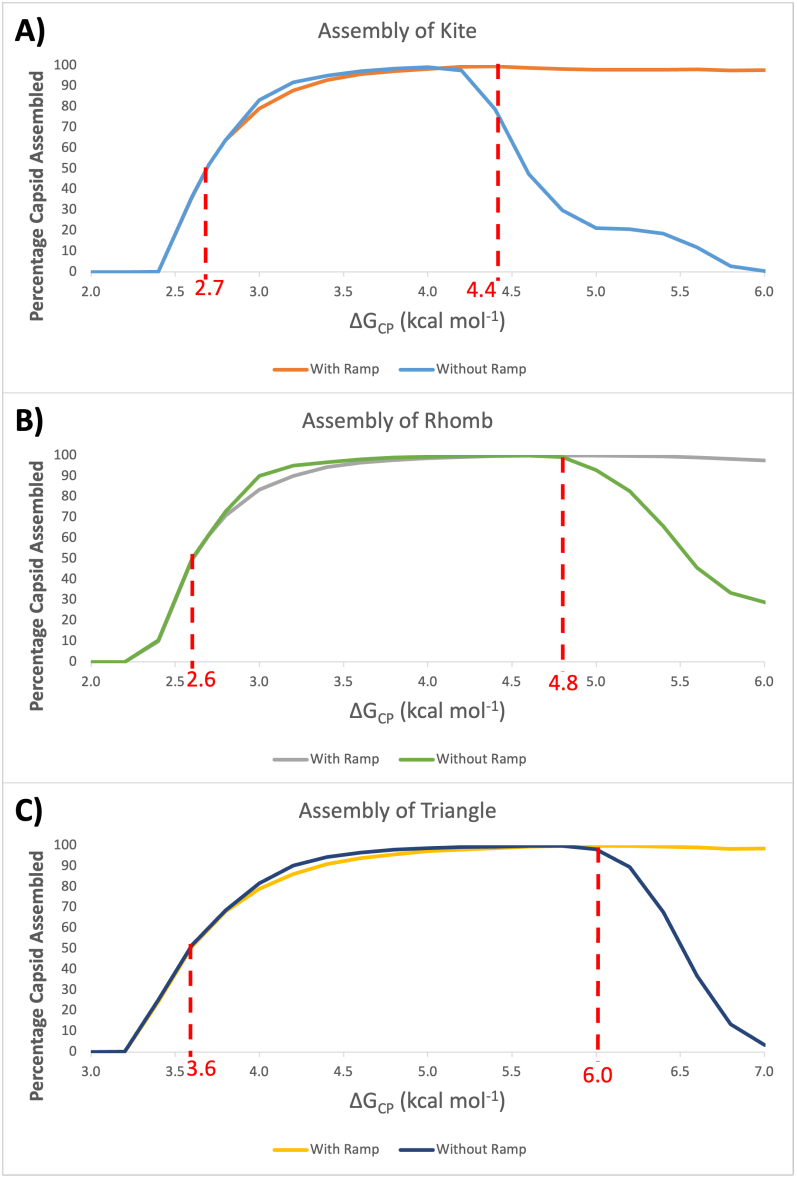

Figure 7.3: Graphs showing the percentage yield from a simulation of 2000 capsid's worth of capsomer, in the absence of RNA, for a variety of ΔG values, starting at $\Delta G_{bond} = 2.0$ (3.0 for Tri) and increasing by steps of 0.2. The ramp, where capsomers slowly accumulate to mimic production from mRNA in a cell by the ribosome, produced capsomers at a rate of 400/s. Without a ramp, the capsomers were all produced instantaneously at the beginning of the simulation. The ΔG_{bond} values at which the ramped simulations achieved their maximal (which was approximately 100%) values (directly from the data, not this graph) were: $\Delta G_{bond}^{Kite} = 4.4$ (kcal/mol), $\Delta G_{bond}^{Rhomb} = 4.8$ (kcal/mol), $\Delta G_{bond}^{Tri} = 6.0$ (kcal/mol). The ΔG_{bond} values for which the ramped simulations found 50% assembly were: $\Delta G_{bond}^{Kite} = 2.7$ (kcal/mol), $\Delta G_{bond}^{Rhomb} = 2.6$ (kcal/mol), $\Delta G_{bond}^{Tri} = 3.6$ (kcal/mol).

2000 nucleated structures, meaning there are too few capsomers available to complete all of the intermediates. As these excess intermediates increase in size, or as their number of them increases, even fewer capsids are able to complete assembly, so the yield decreases.- This is not observed in the simulations where the capsomers are ramped in. This is perhaps due to the production of capsomers being slow compared to the formation of capsomer-capsomer interactions, so that any capsomers produced ended up as part of an existing structure, rather than trying to nucleate into another intermediate, due to the higher stability of pre-existing intermediates.

As can be seen in Figure 7.3, each tiling produces a sigmoidal graph but with a different value for ΔG_{bond} when the assembly becomes efficient. The figures are each labelled with the ΔG_{bond} value where the yield (for the ramped assembly) reached its maximum and also the value where the percentage yield was 50%. The ΔG_{bond} values with the maximal yield were: $\Delta G_{bond}^{Kite} = 4.4$ (kcal/mol), $\Delta G_{bond}^{Rhomb} = 4.8$ (kcal/mol), $\Delta G_{bond}^{Tri} = 6.0$ (kcal/mol). The ΔG_{bond} values where 50% assembly was observed were: $\Delta G_{bond}^{Kite} = 2.7$ (kcal/mol), $\Delta G_{bond}^{Rhomb} = 2.6$ (kcal/mol), $\Delta G_{bond}^{Tri} = 3.6$ (kcal/mol).

The Kite tiling's yield reaches its maximum with the lowest ΔG_{bond} value, closely followed by the Rhomb tiling, with the Tri tiling having a much higher ΔG_{bond} value to reach its maximum. Looking at Table 7.1, this is not the order that would be expected. The Tri tiling has the smallest number of capsomer-capsomer interactions/capsid, so it makes sense that this needs the largest value for ΔG_{bond} however the Rhomb tiling has more interactions than the Kite and yet the Kite can assemble with a lower ΔG_{bond} value (and so with a lower total energy value for the capsid). This suggests that counting the number of interactions is not enough to determine the assembly dynamics. This follows from Chapter 3, which showed how entropic contributions also affect these dynamics.

These maximal yield ΔG_{bond} values are where the three tilings have equivalent assembly outcomes in the absence of RNA. In Section 7.2, these values will be used as an even playing field for the introduction of RNA, so that differences in assembly yield can be attributed to the capsomer-RNA interactions and not just to the number of interactions formed in the capsid intermediate structure.

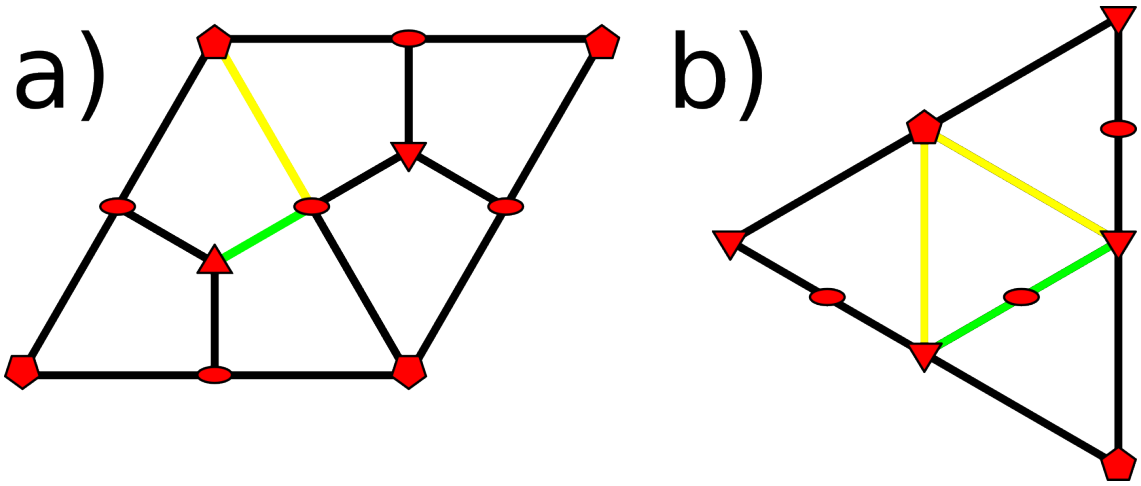

Figure 7.4: The pentagons, triangles and ovals in this figure represent 5-fold, 3-fold and 2-fold axes, respectively. (a) Illustration of the contacts made between a Kite trimer and its neighbouring trimers. Yellow represents the contact around the 5-fold axis (Bond A) and green represents the contact around the 3-fold axis (Bond B). The yellow contact has a larger area than the green contact. (b) Illustration of the contacts made between a Tri trimer and its neighbouring trimers. Yellow represents the contact around the 5-fold axis (Bond A) and green represents the contact around the 2-fold axis (Bond B).

7.1.2 Considering Non-Uniform Capsomer-Capsomer Interaction Strengths

In Section 7.1.1, simulations operated using the assumption that the ΔG_{bond} values of all capsomer-capsomer interactions were the same. This made computation much easier and gave an insight into the approximate values that those parameters should be set to, to ensure an even playing field. However, in a biological context, this is not necessarily a valid assumption. For hydrophobic interactions, the strength of interactions between capsomers is mostly dependent on the buried surface area of the contact. Additionally, when these interactions are dominated by electrostatic interactions, these will not necessarily be uniform across different contact surfaces. For example, looking at the Kite (*e.g.* in Figure 7.4a), the contact area of the interaction about the 5-fold axis is visibly much larger than the contact area of the interaction around the 3-fold axis, suggesting that the interaction about the 5-fold axis should be stronger than the interaction about the 3-fold axis. In Appendix A, this ratio is shown to be ≈ 1.73.

Another example in disfavour of this assumption is MS2, a bacteriophage with a

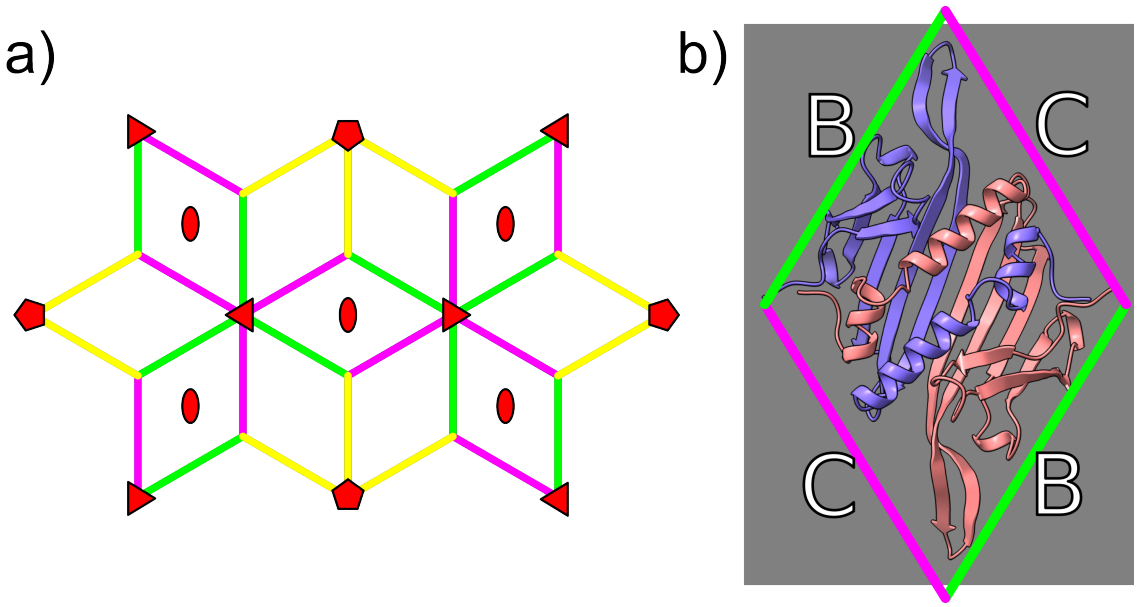

Figure 7.5: (a) Illustration of part of the Rhombic tiling to indicate where the three different boundaries sit between each of the dimers. The boundary about the 5-fold axis (Bond A) is coloured yellow, one of the C-C dimer's boundaries is green (Bond B) and the second (Bond C) is pink. The pentagons, triangles and ovals represent 5-fold, 3-fold and 2-fold axes, respectively. (b) An image of Bacteriophage MS2's C-C dimer, which sits across the 2-fold axis. It is rotationally symmetric but not reflectionally symmetric, so there are two pairs of matching edges, which suggests that any contacts made at these edges will not necessarily have the same interaction strengths. The figure is based on data from ViperDB (ID: 1ZDH) [41]. The boundaries are shown in the same colouring as in (a), to indicate the different boundaries.

Rhombic tiling. It is made of two sets of dimers, where both dimers are made of two copies of the same capsomer but the dimers occur in different conformations. These are called A-B dimers and C-C dimers. An A-B dimer has one protein near a 5-fold axis and the other near a 3-fold axis. A C-C dimer will sit on a 2-fold axis and has both proteins near to 3-fold axes. In drawings of Rhombic tilings (*e.g.* in Figure 7.5a), one would assume that the C-C dimer should have both rotational and reflective symmetries. However, in real viruses they only have the rotational symmetry, as shown by the C-C dimer in Figure 7.5b. Due to this, the C-C dimer will have two pairs of boundaries, where each pair can have different interaction strengths, due to differences in protein structure. In addition to this, there is another boundary that goes around the 5-fold axis, so there are a total of three distinct boundaries, which means three different interaction strengths to vary in this virus.

These are illustrated in Figure 7.5a with each boundary being a different colour.

As a result of these observations, new simulations were run, where the interaction strength was non-uniform. In these simulations, each boundary was given a ΔG_{bond} value and every boundary of that type is set to the same ΔG_{bond} value. During these simulations, the capsomers were added as a ramped process, to mimic *in vivo* assembly. For the Kite and Tri tiling, there are only two different boundaries, so the parameter space is two-dimensional. In the Rhomb tiling, it was initially assumed that all interactions including the C-C dimer were equivalent, so it could be more easily compared to the other tilings. Then, simulations were run, where sufficient capsomers to form 2000 capsids were produced at a rate of 400 capsomers/second and then the simulation was given enough time to reach a stable state. The result of these simulations can be seen in Figure 7.6. In Section 7.1.3, the case without this assumption was investigated, to identify the differences when the C-C dimer's interactions had two different ΔG_{bond} values. Similar simulations were also done in the presence of RNA, covered in Section 7.2.6.

Each tiling has (at least) two distinct capsomer-capsomer interactions, one of which is about the 5-fold axis and the other about another axis. The interaction about the 5-fold axis will be called Bond A and the other interaction will be called Bond B. When considering cases with more than two distinct interactions, these will be clearly defined at the time. Considering Figure 7.6, none of the three heatmaps have symmetry across the line where Bond A and Bond B are equal. The higher yields are found more densely in the regions where one interaction is stronger than the other. All tilings seem to have a preference for Bond B being stronger, to ensure a large yield. Each heatmap has a large plateau, suggesting that whilst the rough values of the interaction strengths are important, they can shift a little without too major an effect on the yield. The Rhomb has the smallest plateau but its plateau has a gentle slope around it, where the efficiency gradually drops away from the plateau. The Kite and the Tri both have larger plateaus but for strengths adjacent to these plateaus, the efficiency drops off very quickly.

During assembly, the nucleation stages tend to be the most unstable, so are likely to follow the path that has the most stable intermediates. For each tiling, making one of the interactions stronger than the other has a positive impact on the assembly yield. When one interaction is stronger, this means that intermediates featuring that interaction will be more favourable. This would favour certain assembly pathways that utilise these more stable intermediates, which seems to lead to a more effective

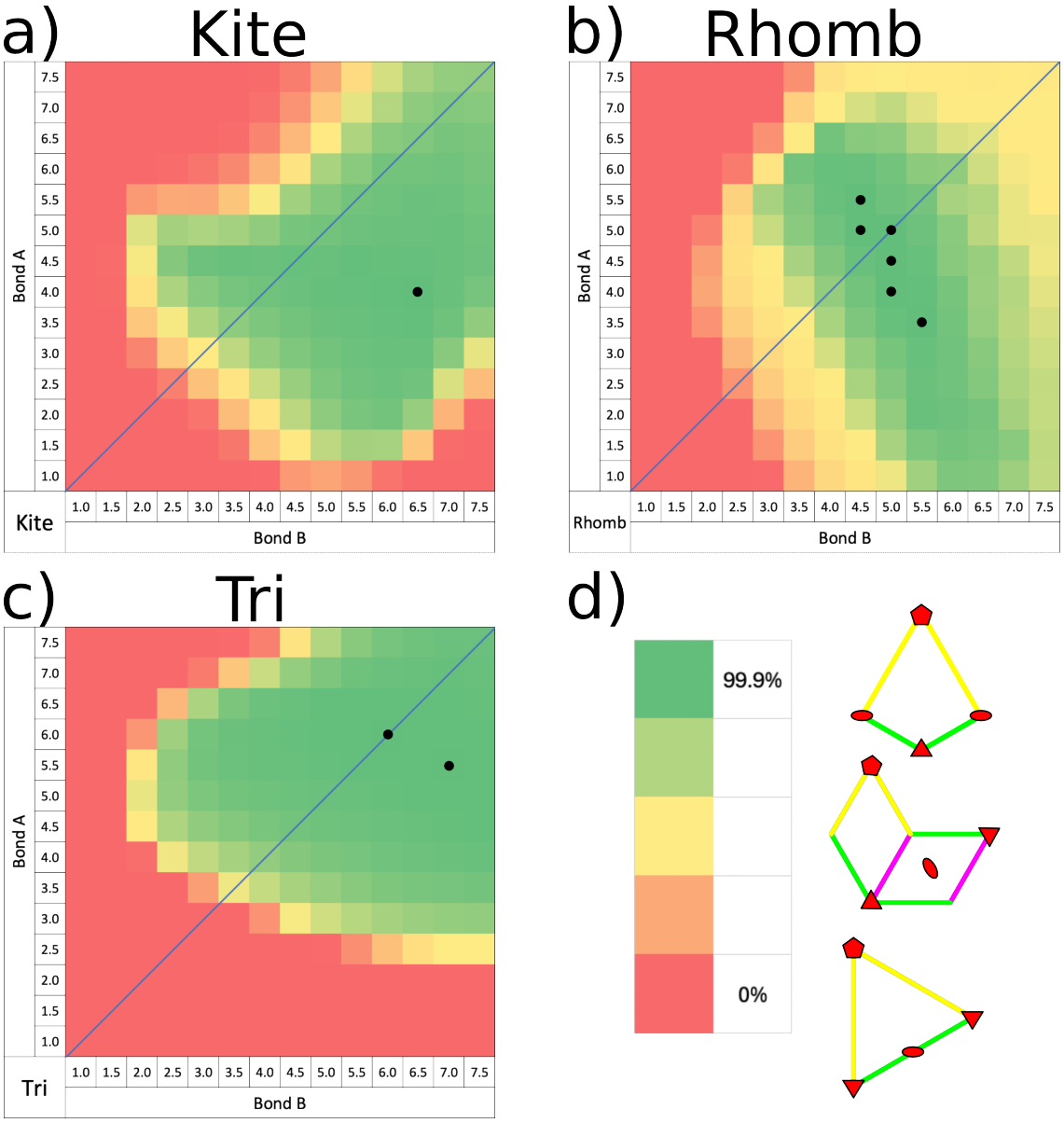

Figure 7.6: Heatmaps showing the relative assembly efficiencies of the three tilings under variation of ΔG_{bond} (measured in kcal/mol) for each boundary, shown for the (a) Kite, (b) Rhomb and (c) Tri tilings. The line indicates where $\Delta G^{A}_{bond} = \Delta G^{B}_{bond}$ and the black dots indicate the maximal values. The scale for all of the heatmaps is given by (d), alongside a diagrammatic representation of the bonds in each tiling. Yellow indicates Bond A, Green indicates Bond B and Pink indicates Bond C, which was set as equal to Bond B in this figure's simulations. In each tiling, Bond A is the interaction about the 5-fold axis. Bond B is the interaction about the 3-fold axis in the Kite tiling, all interactions including a C-C dimer in the Rhomb tiling and the interaction about the 2-fold axis for the Tri tiling. In each heatmap, there is a noticeable plateau (though it is rather small in the Rhomb) that surrounds the values corresponding to maximal yield. In each tiling, varying Bond A and Bond B has different effects on yield, demonstrating that these bonds play different roles in assembly.

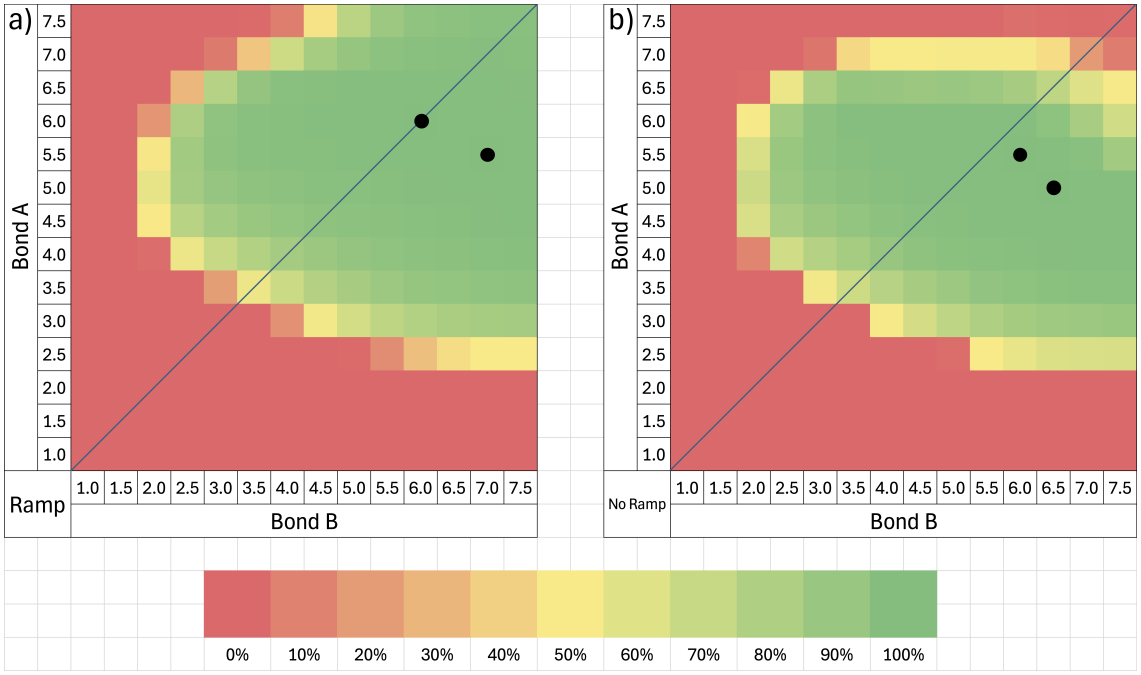

Figure 7.7: Heatmaps showing the relative efficiencies of the Tri tiling under variation of ΔG_{bond} (measured in kcal/mol) for different boundaries. (a) and (b) correspond to simulations with and without a ramp, *i.e.* where proteins are added gradually or all at once, respectively. There is a line indicating where $\Delta G^A_{bond} = \Delta G^B_{bond}$. In both heatmaps, there is a noticeable plateau that surrounds the maximal yield and every point on the plateau has at least 99% yield. The highest points of the heatmap are shown with a black dot. This shows the effect of ramping in the capsomers, where more values of ΔG^A_{bond} and ΔG^B_{bond} are effective than when the capsomers are added all at once.

assembly process. Thus, the Kite tiling will likely maximise the number of times it forms Bond B (about the 3-fold axis), suggesting it initially nucleates using trimers. The Tri tiling favours a stronger interaction about the 2-fold axis (Bond B), so it is likely that dimers play a part in the initial nucleation process for the tiling. It is worth noting however that the capsid has twice as many copies of Bond A than Bond B, which could also play a part. The Rhomb tiling favours the interactions that involve C-C dimers. By favouring Bond B, the assembly would likely begin by forming a hexameric nucleus. This could be as a result of there being twice as many copies of Bond B than there are copies of Bond A but it could also suggest that making sure that C-C dimers are able to easily join the structure is important to the nucleation of the assembly.

To test the effect of ramping in the capsomers, additional simulations generated

the yields when all of the proteins were added at once, rather than gradually. This is shown in Figure 7.7. This makes assembly unfeasible for larger values of ΔG^A_{bond}, potentially, due to an excess of unfinished intermediates that withhold the capsomers needed for other intermediates to complete. Additionally, a lack of ramping seems to make the boundary between the efficient plateau and unassembled capsids sharper. Lastly, the change in shape shown in this range of ΔG_{bond} values suggests that without a ramp, the plateau will be smaller, due to the edges being closer at the boundary of this plot.

Similar work has been done by Wei *et al* [57]. Within this, they approached this problem from two different angles. One was using DNA origami to produce rigid capsomers that are similar to capsomers from a $T = 3$ virus using a Tri tiling, which assemble into an icosahedral structure from 60 capsomers. These rigid capsomers formed attractive interactions via a lock-and-key mechanism. Within these locks and keys, they placed base stacking pairs which formed into "sticky ends", which were responsible for the capsomer-capsomer interactions. By changing the number of these sticky ends, they obtained fine control over the strength of interactions at each boundary between these rigid capsomers. In addition to this, they used a molecular dynamics computational approach to simulate the same system *in silico*. Within both approaches, they ran similar experiments, where they changed the strength of each interaction independently and measured the yield of these assemblies. The graphs of this yield are shown in Figure 7.8 (adapted from their paper [57]). The yield is shown to increase when the two different interaction strengths are different and worsen when they are similar.

This differs from the results that have been shown in this thesis for the Tri tiling. This thesis' assembly efficiencies were higher than theirs, both in their computational and their experimental results. Additionally, the shape of our heatmap of effective assemblies and the shape of their heatmaps were different. Both of their heatmaps had quite narrow values of interaction strength that resulted in effective assembly whereas mine had a larger range of effective interaction strengths that allowed assembly. Their assemblies added all of the capsomers at the start of the assembly and mine used a ramp to add all of the capsomers. Adding capsomers all at once has often reduced the effectiveness of assembly in other simulations. However, Figure 7.7 shows that the effects of a ramp on the assembly is not enough to justify these differences, though some effects (sharper boundaries and inefficient assembly if one ΔG_{bond} is too large) do make the simulations more similar. An unknown factor is the

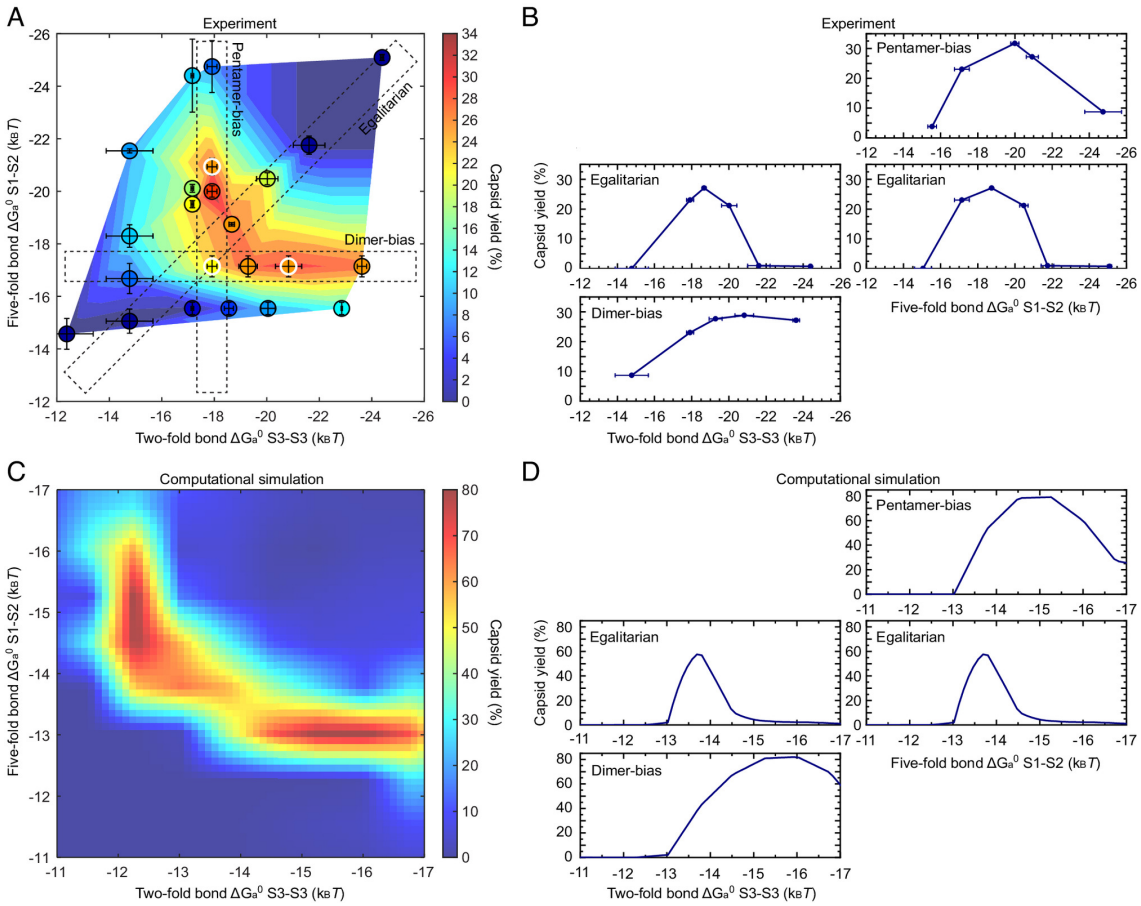

Figure 7.8: Comparison with assembly results by Wei *et al* [57]. Experimental and computational studies of triangular $T = 3$ capsids assemblies. (A,B) feature the yield from their experimental work. These experimental results were obtained using DNA origami to form triangular capsomers and letting them assemble. (C,D) feature the yield from their computational simulations. These computational results are from molecular dynamics simulations of triangular capsomers assembling. Figure adapted from Wei *et al* [57].

volumes used in their simulations and experiments, which could affect the dynamics, so an exact comparison of parameter values with this thesis' work is not feasible. Another difference is that their computational simulation used molecular dynamics, which looks at fewer capsids in each simulation but looks at them more closely. This means it does not rely upon the capsomer graphs that this thesis' model does, so their simulations are able to model defects during assembly, *e.g.* if a capsomer joins the structure at the wrong angle, this could block a capsid's assembly. When the interactions are stronger, these defects are harder to undo, which could justify the difference between the two models when both interactions are strong.

7.1.3 FURTHER NON-UNIFORM INTERACTION STRENGTH IN RHOMB TILINGS

As mentioned in Section 7.1.2 and shown with Figure 7.5, in Rhomb tilings the C-C dimers have two pairs of distinct interactions which do not necessarily have the same strength as each other. As before, Bond A will be the interaction about a 5-fold axis (shown in yellow in Figure 7.5). Bond B is shown in Figure 7.5, as the green boundary and Bond C is shown as the pink boundary.

Therefore, there are three different boundaries in the tiling, meaning an extra dimension to add to the exploration of the parameter space. In Figures 7.9 and 7.10, some heatmaps are shown, where one of the three interaction strengths has been fixed and the other two are allowed to vary.

Initially, we shall consider Figure 7.9, where there are heatmaps comparing Bonds B and C for three fixed values of ΔG^A_{bond}. Data for all values of Bond A from $\Delta G^A_{bond} = 1.0$ to $\Delta G^A_{bond} = 7.5$ in steps of 0.5 was generated. The figure shows three heatmaps, which were chosen due to the trends they show. For $\Delta G^A_{bond} = 1.0$, Bonds B and C need to be quite strong for the capsid to assemble, forming a plateau where both are strong. As we gradually increase ΔG^A_{bond}, this plateau both narrows and also moves both downwards and leftwards until $\Delta G^A_{bond} = 6.0$, when the plateau and the motion changes. Also, after $\Delta G^A_{bond} = 6.0$, the plateau's behaviour reverses, suggesting it has passed the optimal point for ΔG^A_{bond} and that increasing it further worsens the yield. After $\Delta G^A_{bond} = 6.0$, the plateau begins to split into two separate plateaus. This suggests that the capsid needs to have a reaction that is easily reversed, to help prevent kinetic traps.

During this whole analysis, the heatmap has been symmetric about the line where $\Delta G^B_{bond} = \Delta G^C_{bond}$, meaning that changing one interaction's strength has a similar

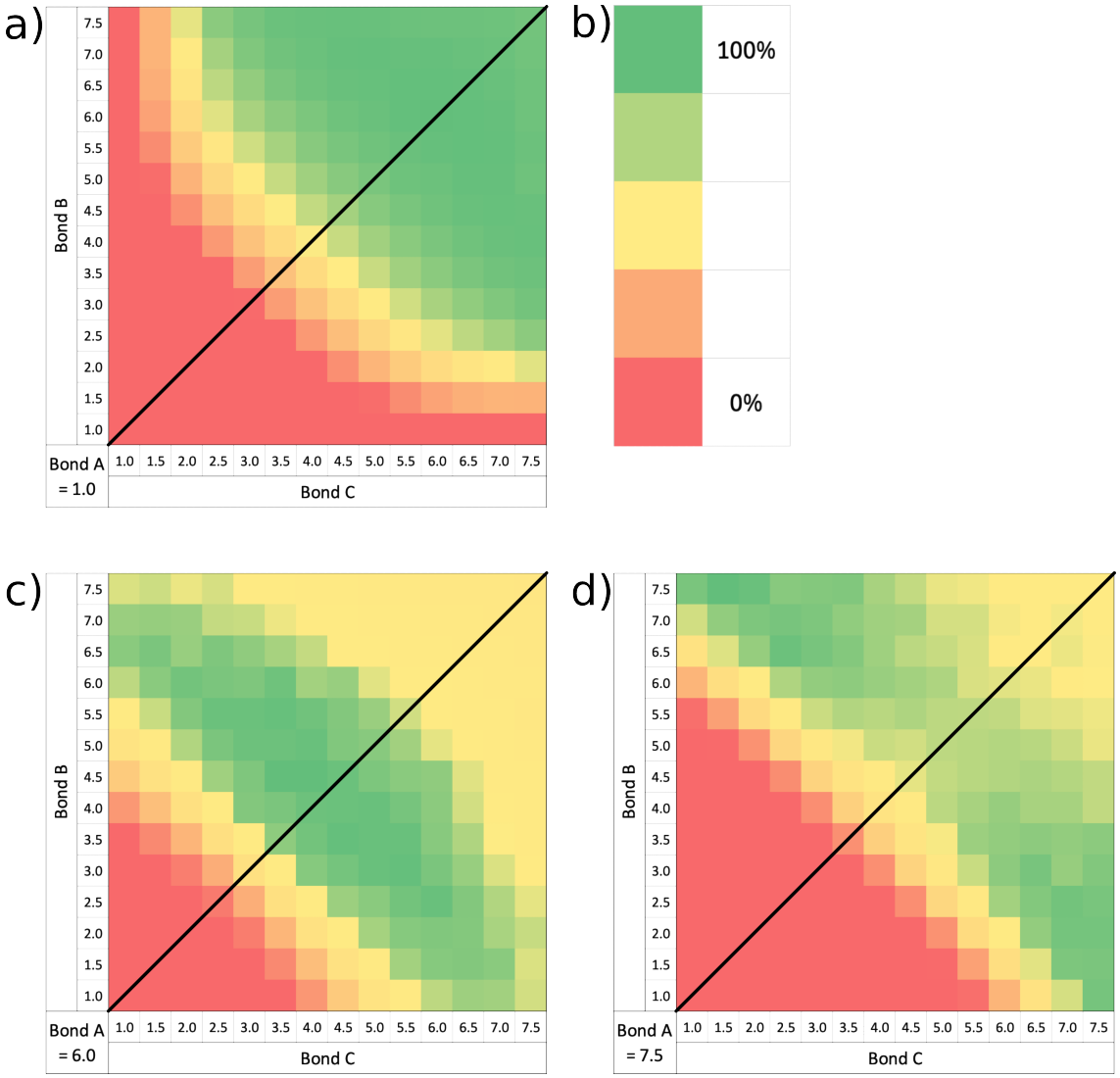

Figure 7.9: Outcomes of assembly simulations for a range of ΔG_{bond} values (measured in kcal/mol) for each of the three distinct boundaries in a Rhombic tiling. The results for three fixed values of ΔG^{A}_{bond} (*i.e.* (a) 1.0, (c) 6.0 and (d) 7.5) are shown here as heatmaps, where the scale is shown in (b). The values of ΔG^{B}_{bond} and ΔG^{C}_{bond} are shown across the vertical and horizontal axes respectively. Each block shown is the average of three data points generated by modelling a system where up to 2000 capsids could assemble. The black lines indicate where $\Delta G^{B}_{bond} = \Delta G^{C}_{bond}$ and the yields are approximately symmetric about this line.

outcome to changing the other. In these graphs, this makes sense, as there is no enforced asymmetry between Bonds B and C and they are reflections of each other. This is not the case in other heatmaps, as Bond A is not equivalent to either Bond B or C. Later, we will see that when RNA is included, this can add in an asymmetry between Bonds B and C, which skews the heatmap, as would be expected.

Next, we consider the heatmaps in Figure 7.10, where it shows three fixed values of ΔG^B_{bond} and variable values of ΔG^A_{bond} and ΔG^C_{bond}. To show differences when compared to Figure 7.9, we will use $\Delta G^B_{bond} = 1.0, 6.0, 7.5$. Unlike in Figure 7.9, there is no branching point in Figure 7.10. The heatmaps show a trend that continues throughout the collected data values for ΔG^B_{bond}, from 1.0 to 7.5. Like in the previous Figure, as the value of ΔG^B_{bond} increases, the plateau with the top yield moves downwards and leftwards.

7.2 ASSEMBLY IN THE PRESENCE OF RNA

Now that assembly of $T = 3$ capsids in the absence of RNA has been modelled, the next step to determine the effect of ΔG_{bond} on assembly is to introduce RNA into these systems. This will show how the presence of RNA impacts on the assembly of the three different geometries and the effects of using similar RNA graphs on different tilings.

One important parameter to consider in these comparisons is the value of ΔG_{bond}, the interaction strength between different capsomers. Initially, choosing this so that each tiling had the same total energy in a completed capsid was considered. Unfortunately, this lead to the Rhombic tiling having too weak an interaction strength to assemble and the Tri tiling having a sufficiently strong interaction strength that most reactions were effectively irreversible, leading to a low yield and only the Kite giving useful insights. Thus, Figure 7.3 was used to determine the value of ΔG_{bond} which gave the highest yield. This resulted in the following values to be used in this chapter, so that this parameter's effects are minimised: $\Delta G^{Kite}_{bond} = 4.4$ kcal/mol, $\Delta G^{Tri}_{bond} = 6.0$ kcal/mol, $\Delta G^{Rhomb}_{bond} = 4.8$ kcal/mol.

The RNAs are assumed to have 60 PSs in each tiling. In the Rhomb tiling, it is assumed that the capsomers which sit on the 2-fold axes do not bind to the RNA and only join the intermediate structure through capsomer-capsomer interactions, as is the case for MS2, a $T = 3$ virus that assembles from Rhombic capsomers.

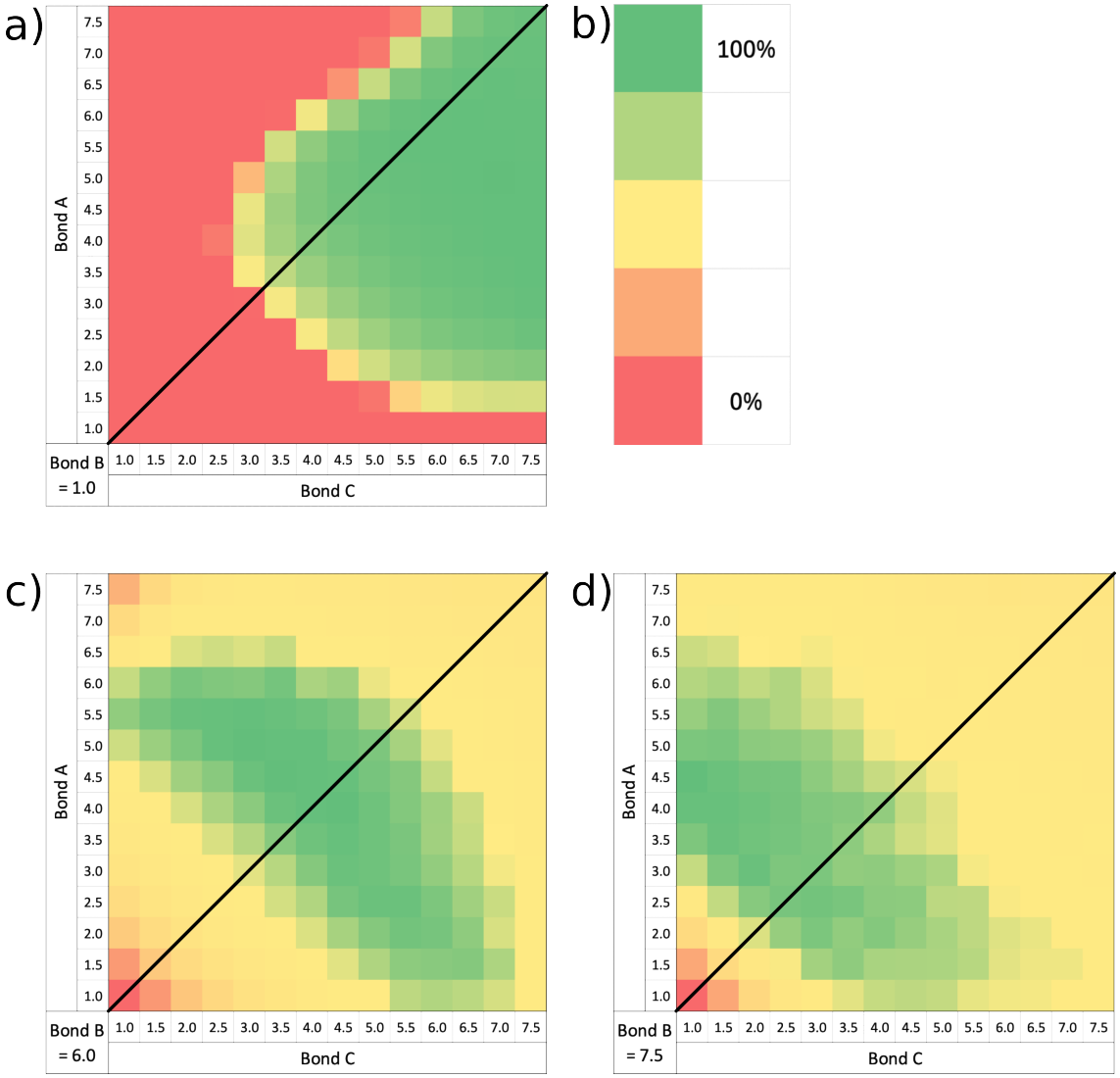

Figure 7.10: Outcomes of assembly simulations for different values of ΔG_{bond} (measured in kcal/mol) for each of the three different boundaries in a Rhombic tiling. The results for three fixed values of ΔG^{B}_{bond} (*i.e.* (a) 1.0, (c) 6.0 and (d) 7.5) are shown here as heatmaps, where the scale is shown in (b). The values of ΔG^{A}_{bond} and ΔG^{C}_{bond} are shown across the vertical and horizontal axes, respectively. Each block shown is the average of three data points generated by modelling a system where up to 2000 capsids could assemble. The black lines indicate where $\Delta G^{A}_{bond} = \Delta G^{C}_{bond}$.

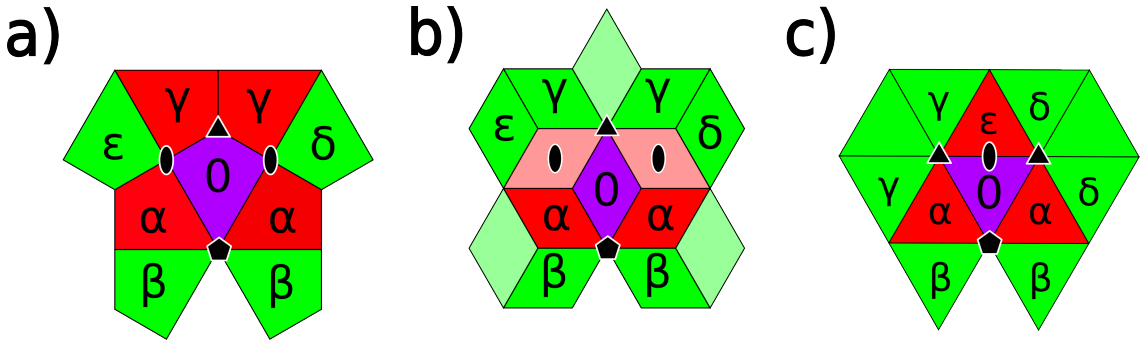

Figure 7.11: Figure indicating the edges on the capsomer graphs and all of the available edges for the RNA graph when using the (a) Kite, (b) Rhomb or (c) Tri tilings. Each (a) kite, (b) rhomb or (c) triangle represents a single capsomer, as shown in Figure 7.1. Red capsomers indicate that these capsomers are adjacent to the purple capsomer in the capsomer graph. Green capsomers are second nearest neighbours. The purple capsomer, labelled 0, is the one the RNA is currently in contact with. When a PS is bound to the purple capsomer, then the capsomers labelled with Greek letters are potential capsomers that an adjacent PS can bind to (*i.e.* $0 \to P$ are edges on the RNA graph from 0 to a point, P). The symmetry axes closest to the purple capsomer are labelled as a pentagon/triangle/oval for 5-/3-/2-fold axes, respectively. In (b), the capsomers that sit on a 2-fold axis are given a paler colour. These capsomers are assumed not to bind to the RNA. To make it simpler to maintain symmetry, the connections that are represented by the same rotation in the opposite direction are labelled with the same letter. These rotations are listed in Table 7.2. Embedding a selection of these edges into Figure 7.1 and drawing edges between capsomers will result in the RNA graph for that assembly scenario. An example of this is in Figure 7.12.

Simulations of the Rhomb tiling where the RNA had 90 PSs were done but these gave very poor yields, so were disregarded.

7.2.1 RNA GRAPHS FOR $T = 3$ CAPSIDS

There are three $T = 3$ capsid tilings: Kite, Rhomb and Tri. A range of different RNA graphs can be constructed for each tiling, based on the symmetry axes near each capsomer. Figure 7.11 shows the available edges for the RNA graphs within each of the three tilings. They are described using rotations in Table 7.2. These edges can be overlaid onto Figure 7.1, to give the full RNA graphs. The edges that can be described as clockwise/anticlockwise rotations by the same angle about the same axis are always paired together and so are given the same label in the $T = 3$ capsids.

Vertex (Kite/Rhomb)	Rotations
α	Rotation about nearest 5-fold axis by $\frac{2\pi}{5}$
β	Rotation about nearest 5-fold axis by $\frac{4\pi}{5}$
γ	Rotation about nearest 3-fold axis by $\frac{2\pi}{3}$
δ	Rotation about one nearby 2-fold axis by π
ϵ	Rotation about the other nearby 2-fold axis by π

Vertex (Tri)	Rotations
α	Rotation about nearest 5-fold axis by $\frac{2\pi}{5}$
β	Rotation about nearest 5-fold axis by $\frac{4\pi}{5}$
γ	Rotation about one nearby 3-fold axis by $\frac{2\pi}{3}$
δ	Rotation about the other nearby 3-fold axis by $\frac{2\pi}{3}$
ϵ	Rotation about the nearby 2-fold axis by π

Table 7.2: A table containing all of the available RNA graph edges, shown in Figure 7.11, expressed as rotations about nearby symmetry axes. Each edge is paired with its inverse, so α represents the rotation in both the clockwise and anticlockwise direction. The Kite and Rhomb tilings both share their nearest symmetry axes, so have been paired here.

A full list of available RNA graphs (produced using every combination of edges listed above) is given in Table 7.3, which gives a numbering scheme indicating both which tiling the graph uses and it can also be used as a reference to which edges are available in that graph. They are labelled using a letter and a number. The letter represents the tiling (K for Kite, T for Tri, R for Rhomb) and the number represents which RNA graph from that tiling is used, as given in the table. To compare two tilings, the table should be referenced - *e.g.* T4 and K4 do not have the same edges but T4 and K6 do. An example of the edges in RNA graph K14 is shown in Figure 7.12 alongside a planar representation of the RNA graph laid onto the Kite tiling. Some RNA graphs are topologically equivalent to other RNA graphs, as can be seen in Figure 7.13. Due to this, all RNA graphs are given an isomorphism type (iso type in the table) and RNA graphs with the same isomorphism type are isomorphic to each other. Thus, differences in assembly within an isomorphism type are due to the differences in the capsomer graph. The table also indicates the mean number of capsids that assembled across three initial assembly simulations of the RNA graphs, featuring 2000 uniform RNAs with $\Delta G_{rna} = 4$ kcal/mol.

Considering the yield from this table, a few interesting observations can be made. Typically, the RNA graphs with low vertex degrees have more RNA graphs with a zero yield, *i.e.* no completed capsids form in this simulation. Additionally, as

Vertex Degree	Rhomb Tiling				Tri Tiling				Kite Tiling			
	RNA Graph	Moves Used	Number Assembled	Iso Type	RNA Graph	Moves Used	Number Assembled	Iso Type	RNA Graph	Moves Used	Number Assembled	Iso Type
3	R1	α, δ	20 ± 4	A	T1	α, ϵ	185 ± 12	A	K1	α, δ	8 ± 4	A
3	R2	β, δ	0 ± 0	B	T2	β, ϵ	0 ± 0	B	K2	β, δ	0 ± 0	B
3	R3	γ, δ	0 ± 0	C	T3	γ, ϵ	0 ± 0	C	K3	γ, δ	0 ± 0	C
3	R4	α, ϵ	25 ± 4	A	T4	δ, ϵ	0 ± 0	C	K4	α, ϵ	5 ± 3	A
3	R5	β, ϵ	0 ± 0	B	-	-	-	-	K5	β, ϵ	0 ± 0	B
3	R6	γ, ϵ	0 ± 0	C	-	-	-	-	K6	γ, ϵ	0 ± 0	C
4	R7	α, β	0 ± 0	D	T5	α, β	0 ± 0	D	K7	α, β	0 ± 0	D
4	R8	α, γ	157 ± 14	E	T6	α, γ	22 ± 8	E	K8	α, γ	74 ± 10	E
4	R9	β, γ	0 ± 0	F	T7	α, δ	21 ± 7	E	K9	β, γ	0 ± 0	F
4	R10	α, δ, ϵ	199 ± 9	G	T8	β, γ	0 ± 0	F	K10	α, δ, ϵ	16 ± 5	G
4	R11	β, δ, ϵ	0 ± 0	H	T9	β, δ	0 ± 0	F	K11	β, δ, ϵ	0 ± 0	H
4	R12	γ, δ, ϵ	0 ± 0	I	T10	γ, δ	0 ± 0	J	K12	γ, δ, ϵ	0 ± 0	I
5	R13	α, β, δ	29 ± 5	K	T11	α, β, ϵ	223 ± 10	K	K13	α, β, δ	8 ± 6	K
5	R14	α, γ, δ	235 ± 20	L	T12	α, γ, ϵ	388 ± 24	L	K14	α, γ, δ	295 ± 14	L
5	R15	β, γ, δ	0 ± 0	M	T13	α, δ, ϵ	387 ± 12	L	K15	β, γ, δ	0 ± 0	M
5	R16	α, β, ϵ	29 ± 3	K	T14	β, γ, ϵ	10 ± 2	M	K16	α, β, ϵ	7 ± 3	K
5	R17	α, γ, ϵ	227 ± 22	L	T15	β, δ, ϵ	8 ± 1	M	K17	α, γ, ϵ	301 ± 14	L
5	R18	β, γ, ϵ	0 ± 0	M	T16	γ, δ, ϵ	230 ± 2	N	K18	β, γ, ϵ	0 ± 0	M
6	R19	α, β, γ	122 ± 10	O	T17	α, β, γ	295 ± 19	O	K19	α, β, γ	133 ± 11	O
6	R20	$\alpha, \beta, \delta, \epsilon$	334 ± 14	P	T18	α, β, δ	308 ± 11	O	K20	$\alpha, \beta, \delta, \epsilon$	89 ± 14	P
6	R21	$\alpha, \gamma, \delta, \epsilon$	413 ± 10	Q	T19	α, γ, δ	325 ± 6	R	K21	$\alpha, \gamma, \delta, \epsilon$	532 ± 14	Q
6	R22	$\beta, \gamma, \delta, \epsilon$	0 ± 0	S	T20	β, γ, δ	0 ± 0	T	K22	$\beta, \gamma, \delta, \epsilon$	0 ± 0	S
7	R23	$\alpha, \beta, \gamma, \delta$	316 ± 23	U	T21	$\alpha, \beta, \gamma, \epsilon$	668 ± 3	U	K23	$\alpha, \beta, \gamma, \delta$	482 ± 17	U
7	R24	$\alpha, \beta, \gamma, \epsilon$	329 ± 8	U	T22	$\alpha, \beta, \delta, \epsilon$	656 ± 11	U	K24	$\alpha, \beta, \gamma, \epsilon$	481 ± 15	U
7	-	-	-	-	T23	$\alpha, \gamma, \delta, \epsilon$	879 ± 8	V	-	-	-	-
7	-	-	-	-	T24	$\beta, \gamma, \delta, \epsilon$	554 ± 11	W	-	-	-	-
8	R25	$\alpha, \beta, \gamma, \delta, \epsilon$	578 ± 21	X	T25	$\alpha, \beta, \gamma, \delta$	876 ± 25	Y	K25	$\alpha, \beta, \gamma, \delta, \epsilon$	888 ± 6	X
9	-	-	-	-	T26	$\alpha, \beta, \gamma, \delta, \epsilon$	1295 ± 11	Z	-	-	-	-

Table 7.3: The numbering scheme used to describe which edges were present in each RNA graph. The three tile types are separated by double vertical lines. Kite tilings are denoted by a K, Rhomb by an R and Tri by a T, followed by a number. The letters $(\alpha, \beta, \gamma, \delta, \epsilon)$ are from Figure 7.11 and represent the subsequent positions that adjacent PSs on the RNA can bind to. Some RNA graphs are isomorphic to others, so there is a column to show this. If the RNA graph for a particular tile has isomorphism type n, the RNA graph is isomorphic to all other graphs with isomorphism type n. Additionally, the assembly yield based off three initial simulations of the assembly of 2000 RNAs per RNA graph, using uniform PS distributions with $\Delta G_{rna} = 4.0$ kcal/mol, is indicated as number of capsids assembled.

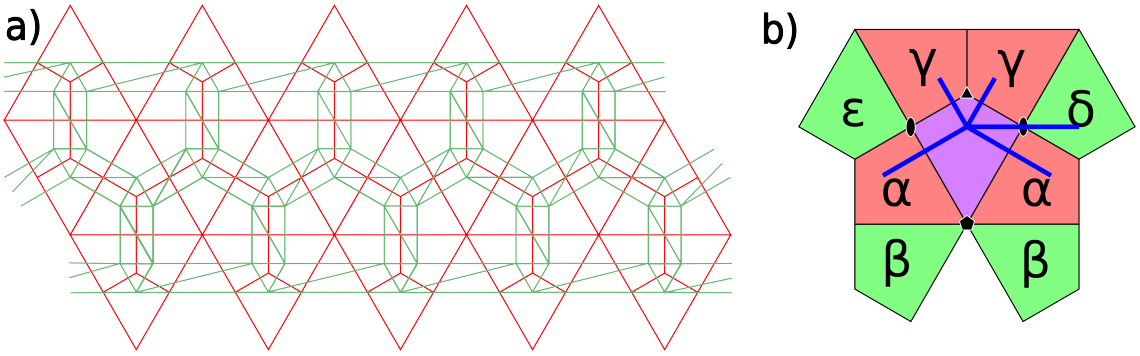

Figure 7.12: (a) Shows RNA graph K14 (detailed in Table 7.3) as a planar net, with the kite shaped capsomers in red and the edges of the RNA graph shown in green. (b) Shows all available edges for Kite tiling's RNA graph, *cf.* from Figure 7.11a. It also indicates the 5 edges chosen for K14, labelled as α, γ and δ, which are shown in blue.

the vertex degree increases, the yield of the assemblies that result in capsids also increases.

The Kite and Rhombic tilings are highly similar in their available edges, such that every RNA graph in one tiling has an equivalent in the other. Thus, this can show interesting differences across a range of the RNA graphs. One initial observation is that Rhombic tilings tend to have a greater yield when the vertex degree is smaller and the Kite tiling generates a larger yield when the vertex degree is higher. The crossover point is where the vertex degree is 6, when K21 is more efficient than R21, R20 is more efficient than K20 and R19 and K19 have similar efficiencies. This supports the notion that vertex degree is not the only factor in the efficiencies and that the geometry of the RNA graph plays an important role, too. This is supported by work by Brunk and Twarock, who also demonstrated the impact of geometry on stability of a capsid [5].

Taking a closer look within certain isomorphism groups, other interesting observations can be found. Whenever comparing two RNA graphs which share an isomorphism group and a tiling, the number assembled is very similar, which is to be expected as the geometries of their capsomer graph and their RNA graph are effectively the same. The only difference between the RNA graphs is them having a different handedness, as shown in Figure 7.13a/b. Typically, the Tri tiling generates a higher yield than the others for isomorphic RNA graphs (with the exception being K8 and R8 both outperforming T6 and T7). Also, the fact that some RNA graphs do not assemble in one tiling does not mean that it will not assemble in other tilings,

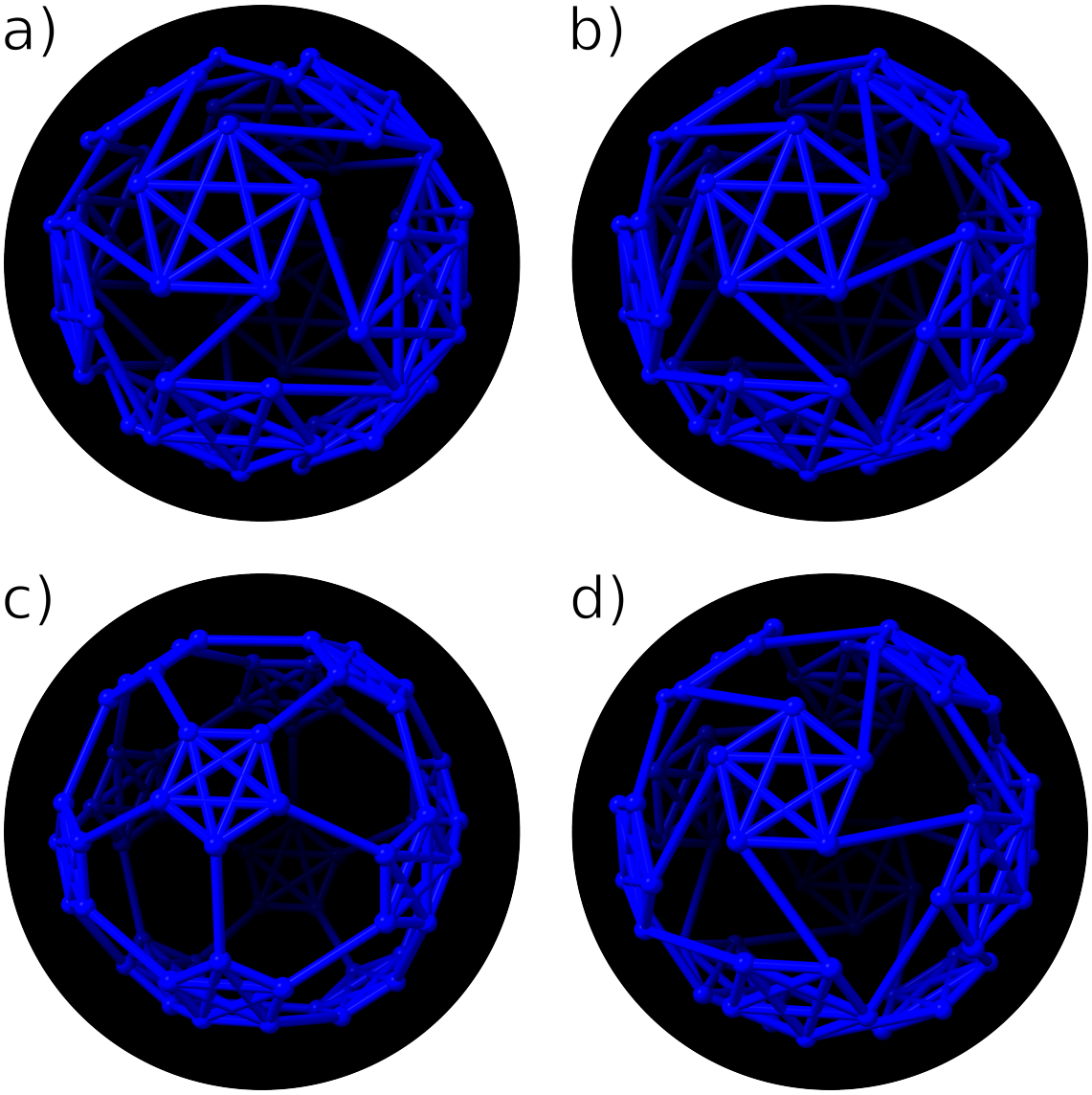

Figure 7.13: A 3D visualisation of four isomorphic RNA graphs. In each case, the nodes (which represent binding sites) are placed approximately at the centre of the capsomer. They represent (a) K13, (b) K16, (c) T11 and (d) R16. These graphs can all be transformed into each other without changing connectivity, via movement of the nodes only. This means that they share an isomorphism group, alongside R13 which is not shown here. The graphs in (a) and (b) are both from the Kite tiling, but show RNA graphs with different handedness.

as shown by K15 and R15 which do not assemble sharing an isomorphism group with T14 and T15 which are able to assemble (albeit somewhat inefficiently).

7.2.2 Optimisation Of The PS-Capsomer Affinity Distribution

In this section, three RNA graphs are chosen to undergo an optimisation procedure, described below, to generate RNAs which assemble with the highest yield. Only three RNA graphs are chosen (one from each tiling), due to the computational time necessary to do this process. These RNA graphs are chosen to test the effects of different capsomer graphs, *i.e.* different tilings, rather than to choose the ones most likely to generate the highest yield. Thus, one RNA graph was chosen within each tiling and all RNA graphs were chosen to be of the same isomorphism type. The three RNA graphs chosen were K14, R14 and T12. During Sections 7.2.2 and 7.2.3, if a Kite/Rhomb/Tri tiling is mentioned, it will refer to this RNA graph as well. These graphs have a better yield than most other RNA graphs with the same vertex degree. The Kite tiling is more effective than the Rhombic tiling, which is not normally the case with smaller vertex degrees, so it is interesting to see if this continues through the optimisation.

The RNA optimisation procedure used here is the same as the one used to optimise the RNA for STNV in Section 6.5. This means 1024 random RNAs are generated, with 60 PSs each (rather than 30, as in STNV). Then the assembly of capsids around these RNAs is simulated. The highest efficiency RNAs were collected and mutated, then the simulations are repeated with these mutated RNAs. This is done until the yield plateaus.

The initial random RNAs were analysed, to test how representative the sampled RNA is, as with STNV. The PSs were binned into high ($8.0 \leq \Delta G_{rna} \leq 12.0$) PSs and low ($4.0 \leq \Delta G_{rna} < 8.0$) PSs. The number of high or low PSs in each RNA was counted, resulting in a binary distribution. Each PS position was observed to have had equivalent opportunities to be low or high too, with the median number of high PSs in each position across RNAs being approximately 512 and all values being within ± 30 of this median.

Midway through the optimisation, we began recording the yield for each RNA at each step in the optimisation procedure. It was estimated that approximately ten optimisation steps had occurred before this. A graphical representation of the 5 best RNAs' efficiency in each step is shown in Figure 7.14. The graph shows

the mean number of assembled capsids and the range of this number for all three tilings. It is based on simulations where 2000 RNAs were present and therefore up to 2000 capsids were able to form. As more optimisation steps were completed, the range of this data got smaller, suggesting that improvements to the top RNAs were stagnating. Additionally, the overall trend became more flat over the final 5 steps for Tri and Kite. The Rhomb tiling showed much smaller improvements than the other two tilings. It also took a lot longer to simulate the assembly and perform optimisation steps, resulting in fewer steps being performed, though it appeared to have started to plateau when its optimisation ended.

To demonstrate the effects on assembly of the optimisation, consider Figure 7.15. This shows a histogram of yields for the initial 1024 RNAs (blue) in comparison with the yield for the 1024 RNAs generated in the final round of optimisation (orange). There is no overlap in the curves for the Kite and the Tri tilings and only minimal overlap for the Rhomb tiling. The initial yield section shows a similar trend in all three tilings: there are a few ineffective RNAs and a few RNAs noticeably more effective than the others but most RNAs sit in the middle of this region. The yield for the RNAs post-optimisation shows skew towards higher yields, which is especially visible in the Kite and Tri tilings but is also visible in the Rhomb tiling.

To finalise the results from the optimisation, the 5 RNAs with the highest yields were identified and their PS distribution was compared to each other. The best RNA generated for each tiling in this RNA graph is shown in Figure 7.16. Each tiling's top RNA was compared to the other top RNAs from that tiling. Each PS position had the mean value of ΔG_{rna} calculated and for Kite and Tri. Over 75% of positions had all of their values across all 5 RNAs within ± 1.0 kcal/mol of this mean. For the ones that did not meet this metric, this was usually due to only one RNA showing a large difference in that position, rather than that position having a large spread of ΔG_{rna} values. This suggests that the optimisation process had concluded and that there were a range of very similar RNAs that gave similar yields, with a low likelihood that any large improvements would be found by continuing to mutate these RNAs. For the Rhomb tiling, there were many more differences between the top 5 RNAs. Only 30% of the positions had all of their ΔG_{rna} values across all 5 RNAs within ± 1.0 kcal/mol of this mean and upon inspection, the values were more widely spread than was observed for the other tilings. This could either mean that more steps of optimisation should have been done, or that the presence of RNA had a lesser effect upon the assembly process.

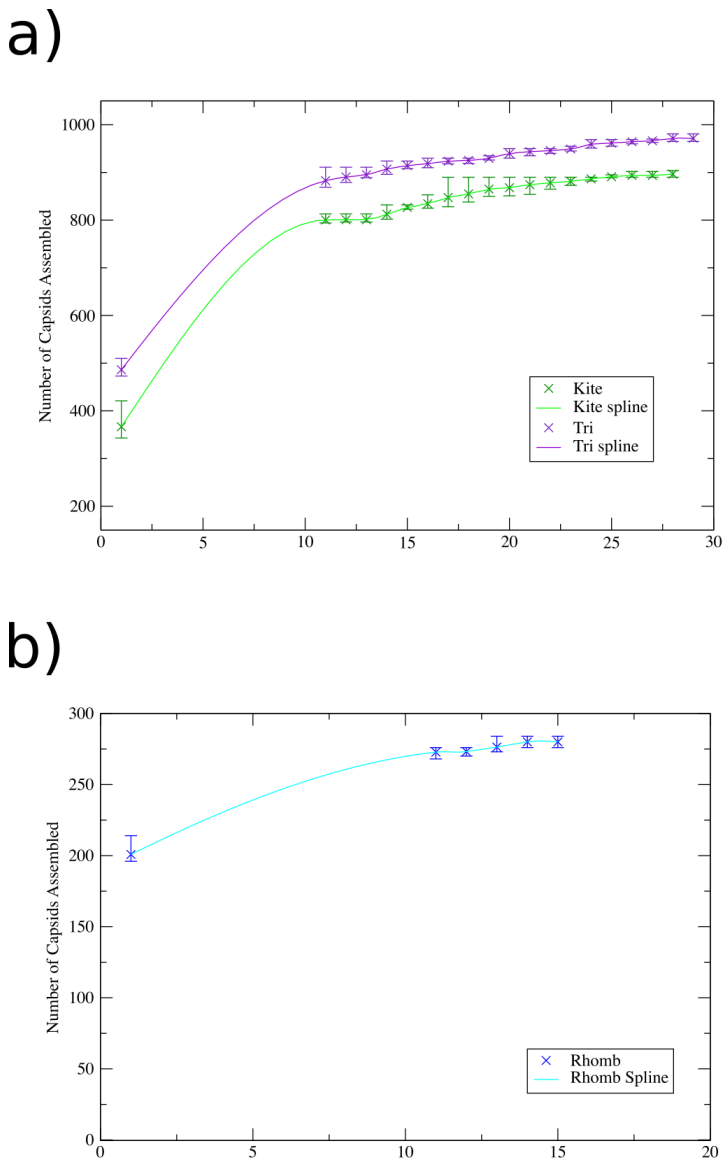

Figure 7.14: After approximately 10 rounds of optimisation, the yields at each step were recorded, to show when the procedure had finished optimising. This can be seen for the Kite and Tri RNA graphs that were being optimised in (a) and it can be seen for the Rhomb RNA graph that was being optimised in (b). These graphs show (for each step in the optimisation process) the mean and the range of the yields of the 5 RNAs with the current highest yield. For the Rhomb, fewer optimisation steps were done, due in part to the longer times needed for each step of the optimisation compared to the other two but also (initially seen before recordings began) the very limited improvements in the yield after each step, which was taken to mean the optimisation was completed.

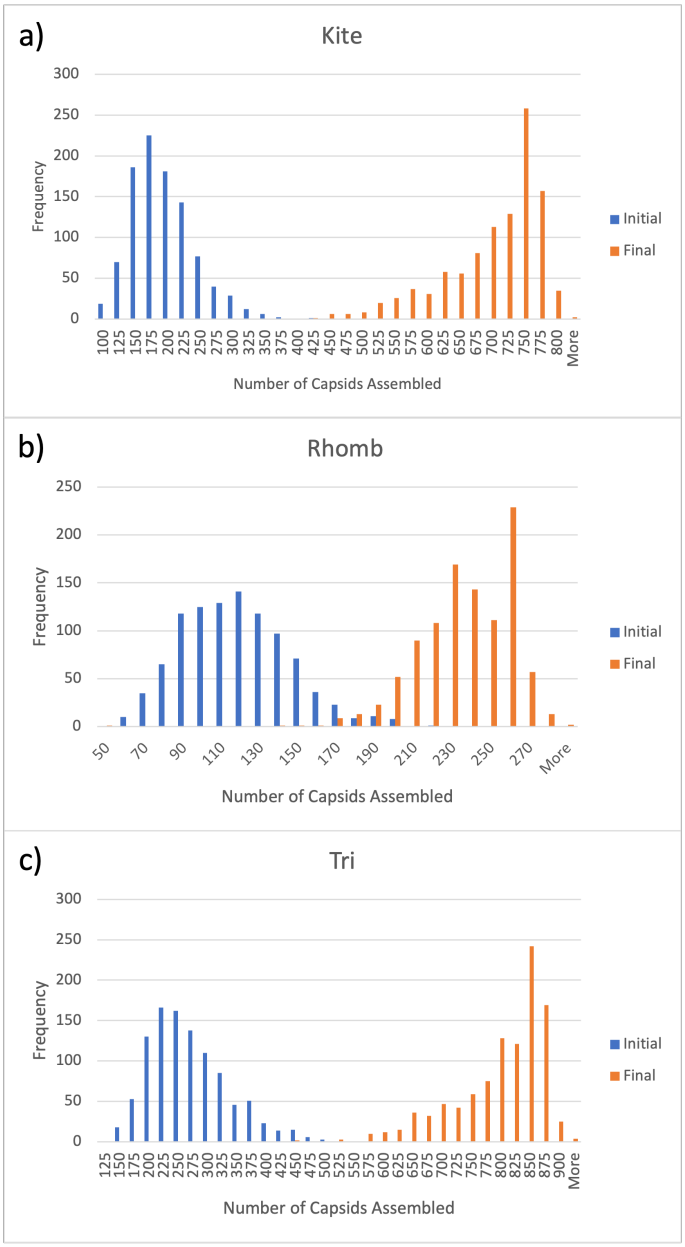

Figure 7.15: After optimisation, the yields from the initial 1024 RNAs and the 1024 RNAs produced in the final step of the optimisation were generated for (a) K14, (b) R14 and (c) T12. In each of these simulations,there were 2000 copies of an RNA and enough capsomers to assemble 2000 capsids. The yields were recorded as number of capsids formed. This data is presented as a histogram, with the initial RNA's yield in blue and the optimised RNA's yield in orange, showing the effect of optimisation on the yield.

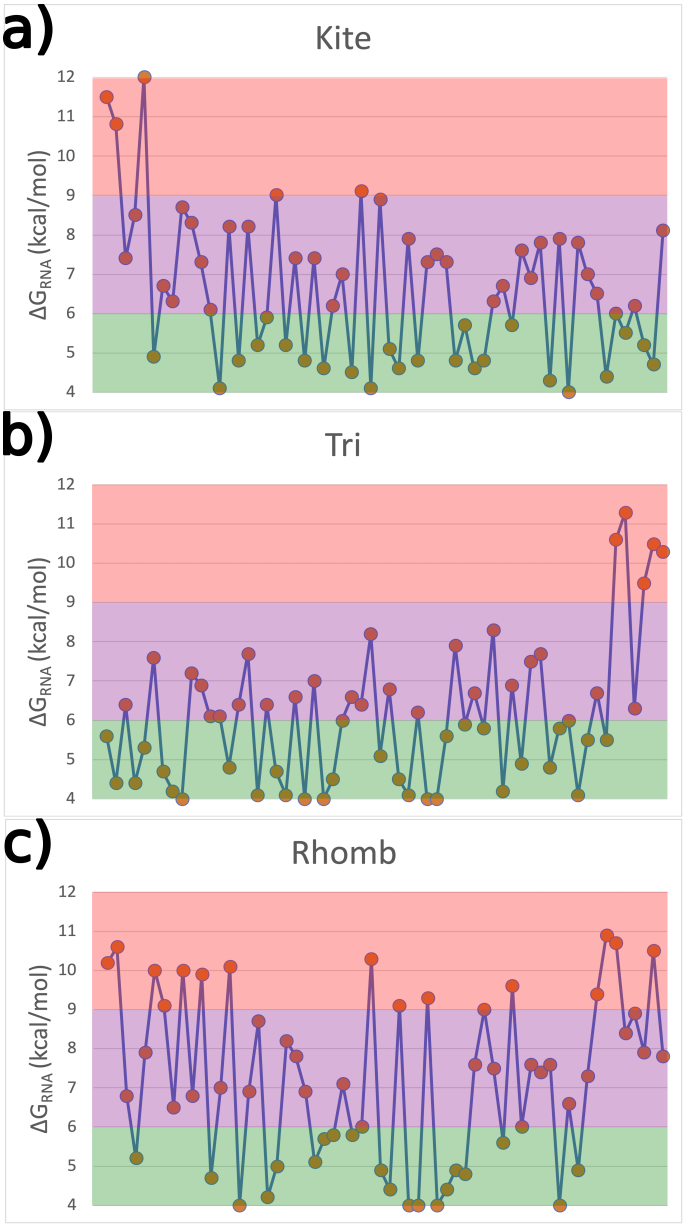

Figure 7.16: At the end of the optimisation, the best RNA for each tiling was recorded and is shown here, where $4 \leq \Delta G_{rna}(n) \leq 12$ kcal/mol for all $n \in \{1, .., 60\}$. The colour boundaries used in Chapter 6 are shown as coloured regions, with red being strongly bonding PSs, purple being medium and green being weak. The top 5 Kite RNAs were highly similar to each other. The top 5 Tri RNAs were also highly similar to each other. Considering Kite and Tri RNAs separately, the mean value for each PS was calculated and the PSs were compared to this mean. Over 75% of PSs had a binding affinity within ± 1 kcal/mol of the mean across all 5 RNAs. The Rhomb tiling did not show this same trend, as only $\approx 30\%$ of PSs fell within this range.

Considering PS distributions of the optimised RNAs in Figure 7.16, there is a certain degree of similarity between the Tri and Kite tilings. Both have a cluster of strongly binding PSs at one end of the RNA, both feature a more weakly binding PS within this cluster. The rest of these RNAs did not feature many strongly binding PSs and had oscillations between weakly binding PSs and medium strength PSs. The Rhombic RNA also showed a lot of oscillations in PS strength but these oscillations had a much greater variance between adjacent ΔG_{rna} values than for the other optimised RNAs. Additionally, there was no clear site where the Rhomb RNA was more likely to nucleate than others.

Some alternative RNAs were tested too and the results can be seen in Table 7.4. These alternative RNAs include a few uniform RNAs (called "XUni", with $\Delta G_{rna} = X$ kcal/mol for all PSs) with a variety of different ΔG_{rna} values and the same uniform RNAs with a nucleation site added at one end (called "XNuc", with $\Delta G_{rna} = X$ kcal/mol in the uniform section and 5 PSs with $\Delta G_{rna} = 12$ kcal/mol).

Generally, if the values of ΔG_{rna} are too high, then this will stop assembly, as shown in the column with a uniform distribution $\Delta G_{rna} = 12$ kcal/mol, where all of the tilings generate less than 1% yield, likely due to kinetic traps and inconsistent nucleation sites. When compared with the equivalent uniform RNA, the yield of RNAs with nucleation sites were always higher, supporting the evidence found in Chapter 6 that a nucleation site is important for effective assembly in this model.

In the AvgNuc column, each tiling has an RNA which uses the average ΔG_{rna} value of the optimised RNA for all of the non-nucleation site PSs. In the Kite and Tri tilings, this had almost no effect on the yield of the assembly compared to the optimised RNAs and it only had a minor effect on the Rhomb tiling. However, the AvgUni column showed a noticeably lower yield for all tilings. Together, this suggests that the sum of the values of ΔG_{rna} across an RNA is less important than the RNA having a nucleation site. In Kite and Tri, AvgUni still had a higher yield than any of the other uniform RNAs, so the total value of ΔG_{rna} across the RNA is still significant.

In these simulations, 4Nuc always had a higher yield than 8Nuc, which suggests that having a lower ΔG_{rna} across an RNA helps assist with assembly, perhaps allowing it to avoid kinetic traps once the capsid has nucleated. In addition to that, 4Nuc's yield ranged from slightly lower than the optimised RNAs' yield to having a higher yield, which suggests this PS distribution is highly effective. When comparing 4Uni and 8Uni, this preference for lower ΔG_{rna} values does not continue. Whilst

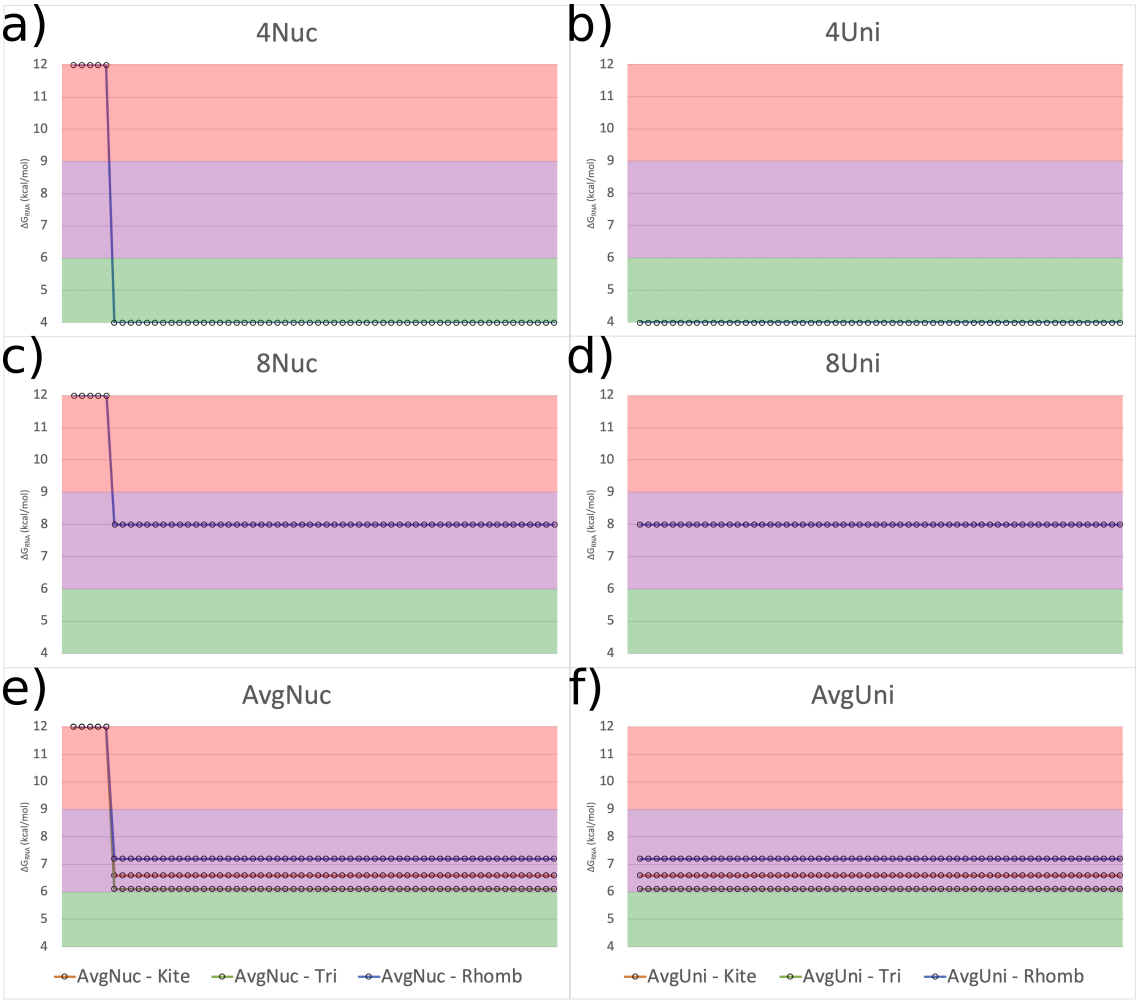

Figure 7.17: The generic RNAs presented in Table 7.4. The "XNuc" RNAs all have a binding site of 5 PSs with $\Delta G_{rna} = 12$ kcal/mol at one end of the RNA. The "XUni" RNAs all have the same ΔG_{rna} value across all PSs. For each tiling, the average values of optimised RNAs' PSs' ΔG_{rna} (as shown in Figure 7.16) was calculated. This gave values of $\Delta G^{avg}_{rna} = 6.6$ kcal/mol for the Kite tiling, $\Delta G^{avg}_{rna} = 6.1$ kcal/mol for the Tri tiling and $\Delta G^{avg}_{rna} = 7.2$ kcal/mol for the Rhomb tiling. The RNAs shown are (a) 4Nuc, (b) 4Uni, (c) 8Nuc, (d) 8Uni, (e) AvgNuc and (f) AvgUni.

Kite							
Optimised	4Nuc	8Nuc	AvgNuc	4Uni	8Uni	AvgUni	12Uni
37.8%	36.2%	32.9%	37.7%	15.0%	17.9%	20.7%	0.1%

Rhomb							
Optimised	4Nuc	8Nuc	AvgNuc	4Uni	8Uni	AvgUni	12Uni
10.8%	20.7%	9.9%	9.6%	11.8%	6.3%	6.0%	0.6%

Tri							
Optimised	4Nuc	8Nuc	AvgNuc	4Uni	8Uni	AvgUni	12Uni
45.0%	54.6%	39.3%	44.8%	18.1%	22.6%	23.6%	0.3%

Table 7.4: Once the optimisation was complete, the RNAs with the best yield for each tiling were identified. Then, simulations of a range of similar RNAs were performed for each tiling, using the same RNA graphs as the optimised RNA. These similar RNAs are shown in Figure 7.17. When these RNAs have uniform ΔG_{rna} values, these will be called "XUni", where $\Delta G_{rna} = X$ kcal/mol. The RNAs with non-uniform ΔG_{rna} in this table are called "XNuc" and all have a nucleation site, with 5 PSs with $\Delta G_{rna} = 12$ kcal/mol at one end of the RNA and the rest of the PSs have uniform $\Delta G_{rna} = X$ kcal/mol across the rest of the RNA. Lastly, each tiling had two RNAs unique to them, where the average value of ΔG_{rna} across the length of its optimised RNA was calculated. This was used to make two RNAs, one uniform, with each PS taking this value of ΔG_{rna} and the other being that same uniform RNA with a nucleation site of 5 PSs with $\Delta G_{rna} = 12$ kcal/mol at one end. This data was generated from one simulation of 2000 RNAs, presented to 1 decimal place.

Rhomb is more effective with 4Uni, Kite and Tri are more effective with 8Uni.

When looking at specific tilings, other noteworthy results can be seen. In the Rhomb table, it shows evidence that the optimisation was not particularly effective for this tiling. Here, two of the generic PS distributions assemble more efficiently than the optimised RNA, with one being almost twice as effective. For Tri, the optimised RNA was better than most of the other RNAs tested but was still beaten by 4Nuc. This shows that an effective RNA was generated by the optimisation procedure but that the whole 60-dimensional parameter space was not fully sampled and that better RNAs may still be out there. For the Kite layout, its optimisation was more successful, giving a higher yield than any other RNA tested here. However, we cannot rule out that there may be more effective PS distributions, due to the size of the parameter space.

7.2.3 Analysis Of Assembly Paths And Intermediates

Once the optimisation was completed, the paths formed during assembly by the optimised RNA were analysed. Initially, this involved the process from 5.2.1, where the paths the RNA traces out on the inside of the capsid were collated and sorted, to tally the frequency of each path. A sample of $\approx 10,000$ paths were generated for each tiling (10022 for Kite, 10130 for Tri and 9393 for Rhomb). When modelling STNV, the capsid assembled with the RNA tracing out the same path in most completed capsids. Repetition of paths was not observed in the Kite, Tri or Rhomb tiling's assemblies with any noteworthy frequency (a small number of paths were observed twice, representing $\approx 0.02\%$ of the sample).

Further investigation into the assembly steps identified interesting features of their assemblies. The methodology is described in Section 5.2.2, where the simulation outputs each reaction that occurs during assembly, so that the assembly intermediates can be identified. A few ways to analyse this were implemented. It is worthy of note (covered in more detail in Section 5.2.2) that due to the simulations running on a Gillespie algorithm, there are a large number of forwards reactions immediately followed by a backwards reactions (or vice-versa), where the net result is no change to the capsid. A filter was put on these reaction, so they were not recorded, as there were usually over 3000 reactions on each RNA when these were included. This will affect some of the data as follows. As so many of these excluded reactions occur, it can be assumed that they are fast reactions and therefore likely to only form/break one bond. In Figure 7.18 (iv), there is a count of how many times a reaction occurs where N bonds are formed or broken, which will be impacted by removing some reactions. Additionally, (iii) in that same figure shows a trajectory of assemblies, which would stagnate more if these reactions were included too.

We start with some basic analysis, as shown in Figure 7.18, presented with comparison with data for STNV. The figure is based on a comparative analysis of the optimised RNAs from Chapter 6. It shows a range of information but the first covered will be from graph (i), which is a histogram containing how many reactions took place to assemble each capsid in that tiling. The Kite tiling has a small tail, in contrast to the Tri tiling. The three $T = 3$ tilings all have a strong skew to the left on these steps but STNV has less skew, with its largest bar being the central one. STNV has a smaller ΔG_{bond} between its capsomers, which could result in more forwards and backwards reactions occurring, perhaps reducing the skew observed in

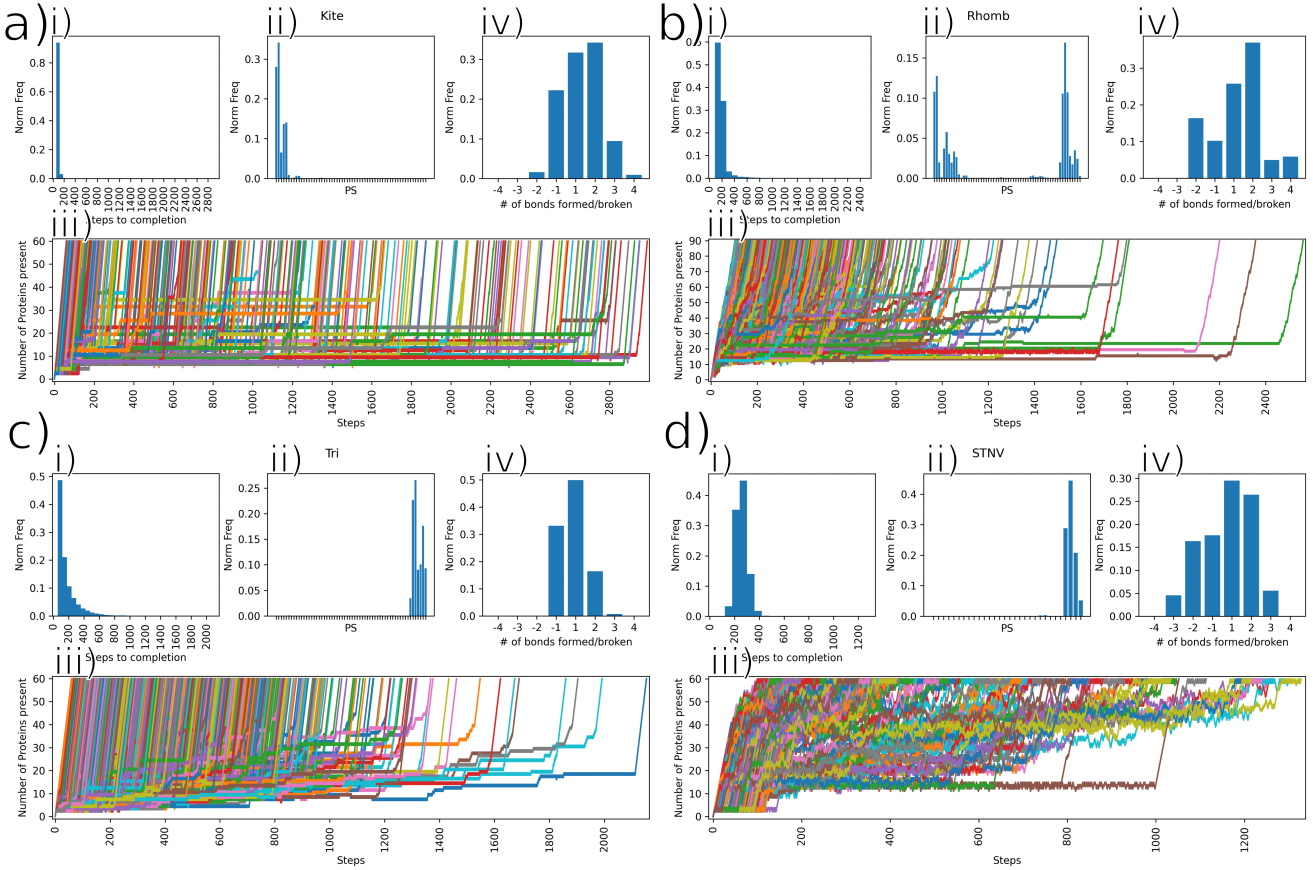

Figure 7.18: Graphs depicting a variety of different features of the assembly for the (a) Kite, (b) Rhomb and (c) Tri tilings alongside the same information for (d) STNV's RNA graph 2345.
(i) A histogram showing the number of reactions before each capsid was fully assembled. (ii) Bar chart illustrating how often the capsid nucleated at each PS. (iii) A line graph showing how the number of capsomers present within each RNA's intermediate structure changes during assembly. (iv) A bar chart indicating how often reactions which cause a net change in the total number of capsomer-capsomer contacts occur. Positive numbers indicate these contacts forming and negative indicates these contacts breaking. These graphs were generated using data from approximately 10,000 assembled capsids for each tiling.

the $T = 3$ capsids.

In (ii), the sites that each tiling nucleates at most frequently are shown. This means that there are two adjacent PSs, each with a capsomer bound to it and then these two capsomers associate with to each other, thus starting formation of the intermediate structure. As is expected, the most common nucleation sites are the strongly binding PSs at one end of the RNA. The Rhomb tiling's optimised RNA does not have a clear nucleation site, as it nucleated in a wider range of positions. However, it still seems to preferentially assemble from one end of the RNA or the other, instead of starting the assembly mid-way along the RNA, which is an interesting observation.

Graph (iii) shows how many capsomers were in each RNA's intermediate structure after a given number of reactions. This roughly represents how an intermediate's size changes over time, though the time axis would not be linear. Vertical lines indicate that the reactions are either consistently forwards or consistently backwards (typically just forwards). Horizontal lines indicate that there are a large number of forwards/backwards reactions in sequence, suggesting that the assembly has stagnated at this position or is potentially stuck in a kinetic trap (though not an effective one as all trajectories shown here are for capsids that completed assembly, so it will have escaped). This graph has its axes match the graph above it, which indicates where most of the lines end. In the $T = 3$ viruses, most of the trajectories are nearly vertical, so the assembly proceeds swiftly to an assembled capsid. There is a limited amount of horizontal lines, which were predominantly observed with smaller sized intermediates, showing that once these $T = 3$ tilings had a certain threshold amount of capsomers present, their assembly did not stagnate. STNV had a lot more horizontal lines visible and showed a lot more backwards reactions than the others. This is potentially due to capsomers that do not bind to the RNA joining its capsid, as without the stabilising effect of the RNA, they can leave the structure easily. However, if this was the only reasons, then the Rhomb trajectories should also show this effect, so another explanation is needed. Another reason is that STNV has a lower ΔG_{bond} than any of the $T = 3$ capsids, so the intermediates were less stable.

Finally, graph (iv) shows the relative frequencies of reactions that formed or broke a given number of capsomer-capsomer interactions. This will be skewed away from the true value, as a number of forwards/backwards reactions were filtered out (specifically, when the net result of two consecutive reactions was no change, these reactions

were not recorded). However, due to the frequency of these reactions, they are likely the faster reactions, forming/breaking smaller numbers of capsomer-capsomer interactions. None of the $T = 3$ capsids have any visible reactions that break three or more capsomer-capsomer interactions, meaning that these occur extremely infrequently, if at all. The vast majority of reactions counted are either breaking one interaction, or forming either one or two interactions. As these simulations all result in assembled capsids and begin with free capsomers, there is obviously going to be a skew towards formation of these interactions, rather than breaking them. STNV does involve reactions where three interactions are broken but this is again likely due to the lower ΔG_{bond} value when compared with the $T = 3$ capsids.

Following this, code (as described in detail in Section 5.2.2), utilised the data in Figure 7.18 to create a list of the last intermediate of each size for a given RNA's capsid assembly. Two scenarios were considered: one where two intermediates match if their capsomer and RNA graphs' occupancies are equivalent, modulo rotational symmetry. The results for this variant will be discussed here. The second variant only looked at the capsomer level, so considered two intermediates equivalent if only the same capsomer structure was present; the results from this analysis are covered in Section 7.2.4.

Throughout searches of smaller intermediates, to identify any similarities of RNA paths, no significant alignment ($\geq 1\%$ of intermediates match) was found for intermediates including 15 or more capsomers. When considering smaller intermediates, such as those including 5 capsomers, there was some correlation in the RNA paths. However, this was at most 10% per path for the most common intermediates. Using RNA graphs with 5 edges per vertex and intermediates with 5 capsomers, there are 4 edges in this path, leading to a maximum of $5 * (5 - 1)^3 = 320$ different paths the RNA could take. This is without even considering the necessary capsomer interactions, so some overlap is both expected and likely. For paths that include 60 capsomers, the maximal number of paths is much larger, so repetition would only occur if there was a thermodynamic advantage to it. No repetition of a path more than twice over $\approx 10,000$ sampled paths was observed. This suggests that for the RNA graphs chosen within the three $T = 3$ tilings, there was no thermodynamic advantage for choosing a single path and optimising the RNA so that this path would always form. An alternative argument would be that as more PSs are added to the RNA, the space of potential paths increases exponentially. Due to the huge space of potential paths, it is noteworthy that any path is observed multiple times, as it has

numerous alternative options. As a result, this might suggest that this method of simulating viral assembly, as a series of biochemical reactions on graphs with RNAs that have a fixed number of PSs, has limited effectiveness within larger viruses.

An alternative argument would be that, as demonstrated for STNV, not all RNA graphs will show repetition of paths after optimisation is run. From this, it could be concluded that the RNA graph choices made here were simply not RNA graphs that would result in paths being repeated and that were the optimisation procedure to be repeated with more RNA graphs, repeated paths may be observed.

7.2.4 Intermediates Bar Chart

As the optimisation procedure did not result in PS distributions which assembled with the RNA tracing out the same path within the capsid, a focus was placed on the shape of the intermediates and where the capsomers sat within a range of intermediates. The code defined in Section 5.2.2 was implemented and the latest intermediates of each size for each RNA were identified. Then, they were grouped to see how often each intermediate was present within these simulations. This section will cover this data, considering all RNA graphs defined in Table 7.3 and two different choices of RNA used for each RNA graph. Due to the capsomers that can join the capsid without being bound to RNA being less stable and their more frequent addition/removal interfering with the results for the Rhomb tiling, this analysis was only applied to the Kite and Tri tilings. To begin, how these RNAs were chosen is described.

As is shown in Sections 7.2.2 and 6.5, the optimisation of RNA typically ended up generating a site on the RNA (often at one end) that has a series of strongly binding PSs. As in this section we will be reviewing a range of RNA graphs, two RNAs were used for each RNA graph. One RNA was a uniform RNA with $\Delta G_{rna} = 4$ kcal/mol and the other was mostly uniform ($\Delta G_{rna} = 4$ kcal/mol) with a cluster of 5 stronger PSs ($\Delta G_{rna} = 12$ kcal/mol) at one end. These two RNAs are shown in Figure 7.19, termed uniform RNAs and nucleating RNAs, respectively. Typically, the nucleating RNA results in a better yield, as is shown in Table 7.5, so the sample size for the nucleating RNAs' graphs will be slightly larger.

Table 7.5 shows the RNA graphs that were analysed and how many capsids assembled during a single simulation of 2000 RNAs for each of the RNAs. As was observed above, adding a nucleation site onto the uniform RNA significantly

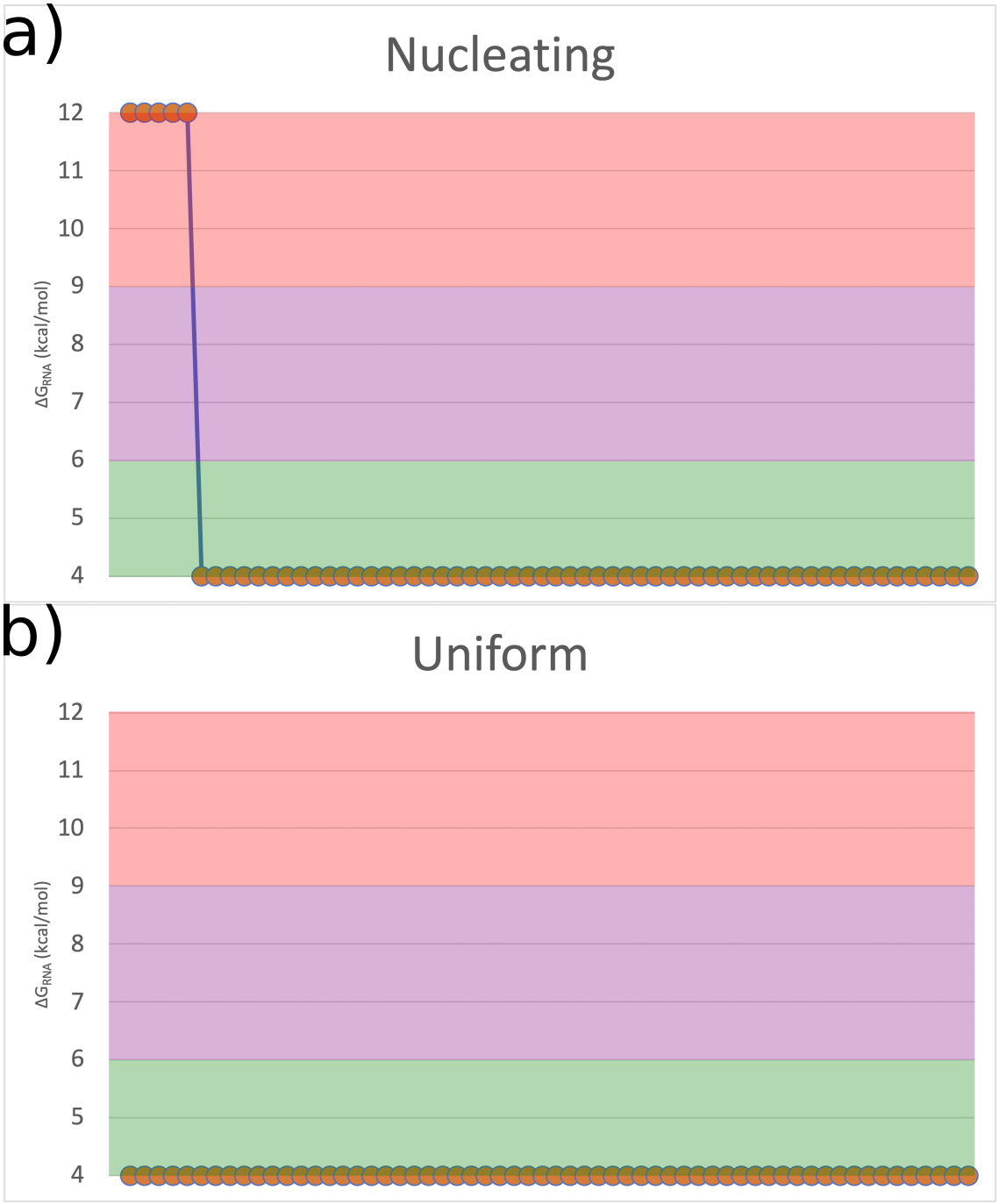

Figure 7.19: Figure indicating the ΔG_{rna} values for the two generic RNAs used in assembly simulations in Section 7.2.4. One has the same ΔG_{rna} value for all of its PSs, so is termed the Uniform RNA. The other is the same with a nucleation site added to one end, so is termed the Nucleating RNA. The colouring scheme is the same as in Figures 6.10 and 7.16.

Vertex Degree	Tri Tiling				Kite Tiling			
	RNA Graph	Iso Type	Uniform Yield	Nucleating Yield	RNA Graph	Iso Type	Uniform Yield	Nucleating Yield
3	T1	A	340	1063	K1	A	16	129
3	T2	B	0	0	K2	B	0	0
3	T3	C	0	0	K3	C	0	0
3	T4	C	0	0	K4	A	8	129
3	-	-	-	-	K5	B	0	0
3	-	-	-	-	K6	C	0	0
4	T5	D	0	0	K7	D	0	0
4	T6	E	39	30	K8	E	112	439
4	T7	E	31	24	K9	F	0	0
4	T8	F	0	0	K10	G	18	10
4	T9	F	0	0	K11	H	0	0
4	T10	J	0	0	K12	I	0	0
5	T11	K	429	875	K13	K	7	4
5	T12	L	624	1485	K14	L	366	1071
5	T13	L	584	1514	K15	M	0	0
5	T14	M	38	27	K16	K	7	4
5	T15	M	35	15	K17	L	344	1046
5	T16	N	314	656	K18	M	0	0
6	T17	O	443	870	K19	O	188	771
6	T18	O	424	901	K20	P	94	75
6	T19	R	523	572	K21	Q	627	1511
6	T20	T	0	0	K22	S	0	0
7	T21	U	994	1765	K23	U	575	1355
7	T22	U	1047	1767	K24	U	562	1299
7	T23	V	1228	1759	-	-	-	-
7	T24	W	729	1042	-	-	-	-
8	T25	Y	1196	1405	K25	X	936	1735
9	T26	Z	1598	1965	-	-	-	-

Table 7.5: Table showing the number of capsids assembled for each RNA graph, which represents the sample size of the data used in Section 7.2.4's analysis. These simulations include 2000 RNAs, so could form up to that many capsids and are performed for two RNAs. One with a uniform PS distribution where $\Delta G_{rna} = 4$ kcal/mol for all PSs and another with a nucleation site of 5 PSs with $\Delta G_{rna} = 12$ kcal/mol at one end (see Figure 7.19). The isomorphism types from Table 7.3 are also given. As this analysis was not applied to the Rhombic tiling, it is not featured here.

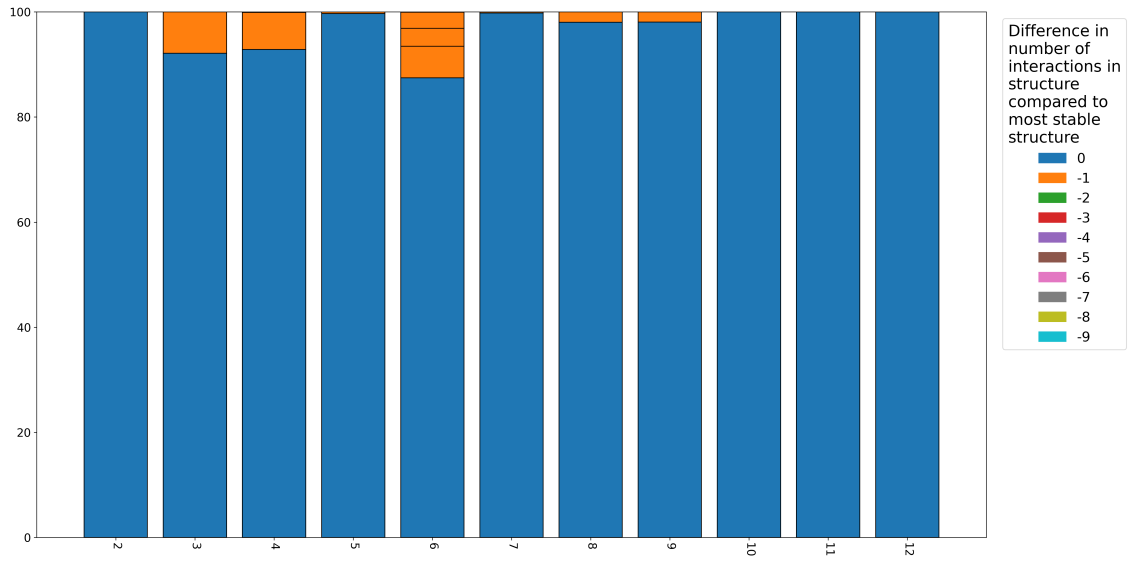

Figure 7.20: Assembly of a dodecahedral capsid, made from 12 pentameric capsomers, using the optimal RNA defined in work by Dykeman *et al* [19]. As described in Section 5.2.2, the latest intermediate of each size was recorded and then these intermediates were grouped together and counted. Each column of bars shows a separate intermediate size and each bar shows a different intermediate and the percentage of the latest intermediates that form it. The bars are sorted so that the most common intermediates are the lowest bars. They are colour coded, with the legend showing the difference in the number of capsomer-capsomer interactions between the intermediate with the largest number of capsomer-capsomer interactions and that bar. This shows that most of the intermediates occurring in the simulations are the most stable intermediates available.

increased the total number of assembled capsids. In a small number of cases, it did not improve the yield. However all of these cases had a yield of less than 100 capsids, regardless of whether there was a nucleation site or not.

Before looking in detail at bar charts for $T = 3$ capsids, a smaller example will first be examined. Figure 7.20 shows a bar chart including the stabilities of the intermediates formed during the assembly of a dodecahedral capsid, made up from 12 pentameric capsomers. Each column represents the intermediates featuring a given number of capsomers present. Each bar represents the frequency of assemblies that feature a given intermediate. The bars are colour coded to indicate the stability of the intermediate, with blue being the most stable observed intermediates (*i.e.* the ones with the largest number of capsomer-capsomer interactions) and a sliding scale for the intermediates with fewer and fewer interactions.

In this example, most columns have only one bar, suggesting that most intermediates correspond to the most stable intermediate. Even the column with the smallest blue area has approximately 90% of the observed intermediates being blue. There are no green bars shown, so no intermediates occur with 2 fewer interactions than the best. This validates some of the assumptions used in Chapter 3, as these stochastic simulations of dodecahedral capsids predominantly use the most stable intermediates available.

Some of the sample sizes in Table 7.5 are quite small, which leads to the question of how large should the sample be to get a representative result. Thus, Figure 7.21 was derived. A simulation of RNA graph K14 using 2000 copies of its optimised RNA generated 936 sampled capsids. Using this data, various sample sizes were taken ($a \times 10^b$, for $a \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $b \in \{1, 2\}$, alongside one featuring all of the data). Then the graphs for these samples were formed and they were compared to the graph for the full sample. To some degree, all of the graphs had some resemblance to the full sample's graph but the smallest sample size that showed a significant similarity was 50 capsids (Figure 7.21a), in contrast with the graph for the full sample (Figure 7.21b).

The first similarity to note here is the general shape of the non-blue section of the chart, with peaks and troughs at the same positions. Looking more closely, with both smaller and larger intermediates, there are spikes of orange (intermediates with one fewer capsomer-capsomer interactions than the most stable bars in each column) at the same positions in the chart. Next, there are five columns where there is a significant amount of green bars present (which contain fewer capsomer-capsomer interactions than orange bars), typically in regions where less of the column was taken up by blue bars. These green areas are at the same positions in both charts and have orange areas nearby. There are some cases where they are not equivalent, such as the extra orange bar in the middle of blue in the smaller sample and another column of mostly green bars in column 17, that is not well reflected in the larger sample's graph. This is likely due to fluctuations resulting from the smaller sample size. The black region at the top of the second graph is due to a large density of very thin bars, so only the lines between them can still be made out. In this region, the smaller sample has many bars that are based on just one intermediate, so the two graphs are consistent here.

To conclude, the chart with a smaller sample is not exactly the same as the chart with the larger sample, so obtaining a larger sample is still necessary to get the

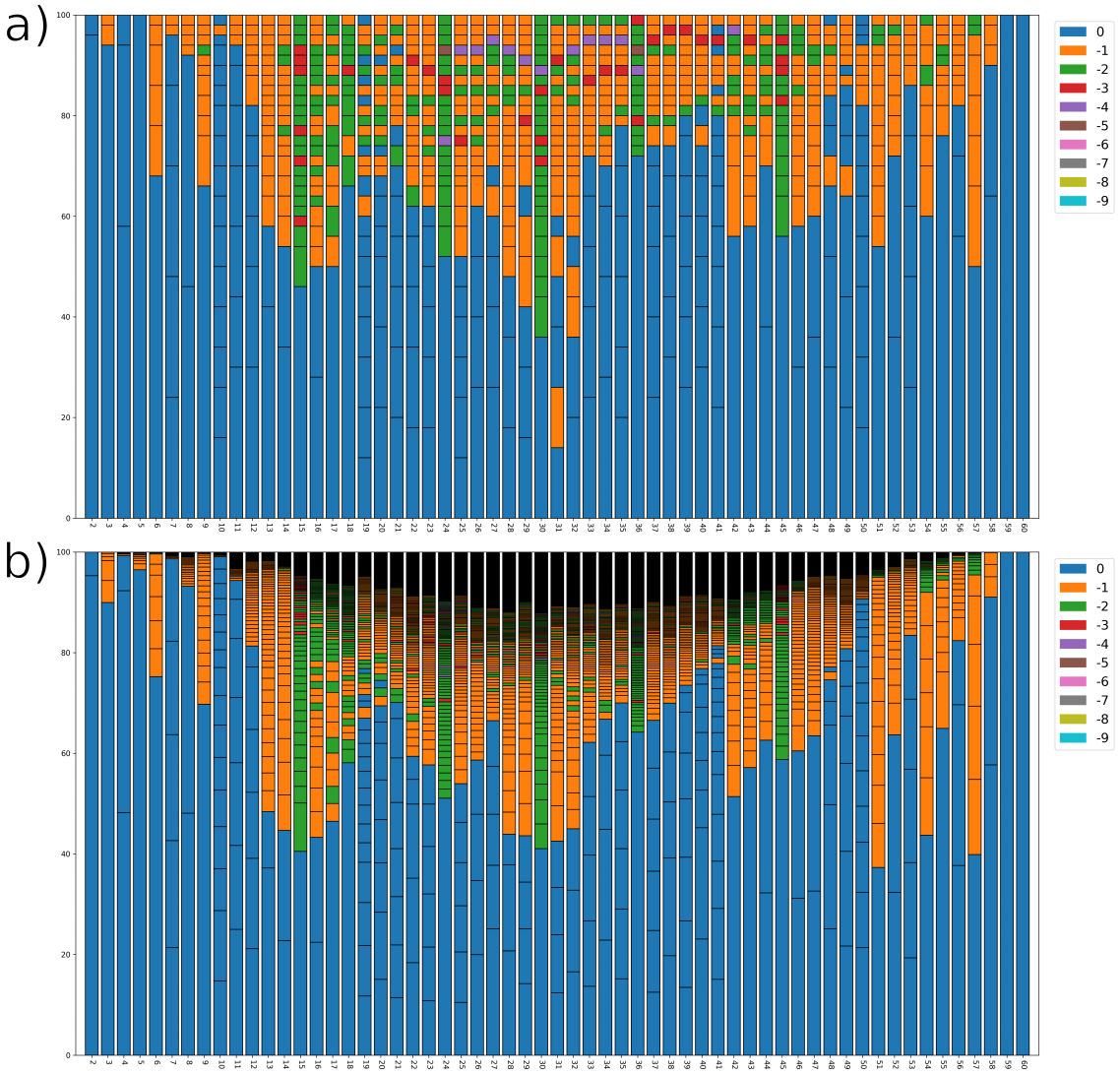

Figure 7.21: The latest intermediates of each size during simulations were recorded for a simulation of RNA graph K14 using its optimised RNA. The colour scale indicates the difference between the number of interactions formed in an assembled intermediate structure and the number formed in the most stable structure of that size, with the key shown on the right. Then, different subsets of these intermediates were analysed, to see how large the sample of intermediates needs to be to reflect a large sample size. The full sample generated included over 900 intermediates but even a sample as small as 50 will give an output which shows similar trends to the full samples. These two graphs show a sample of (a) 50 capsids and (b) 936 capsids.

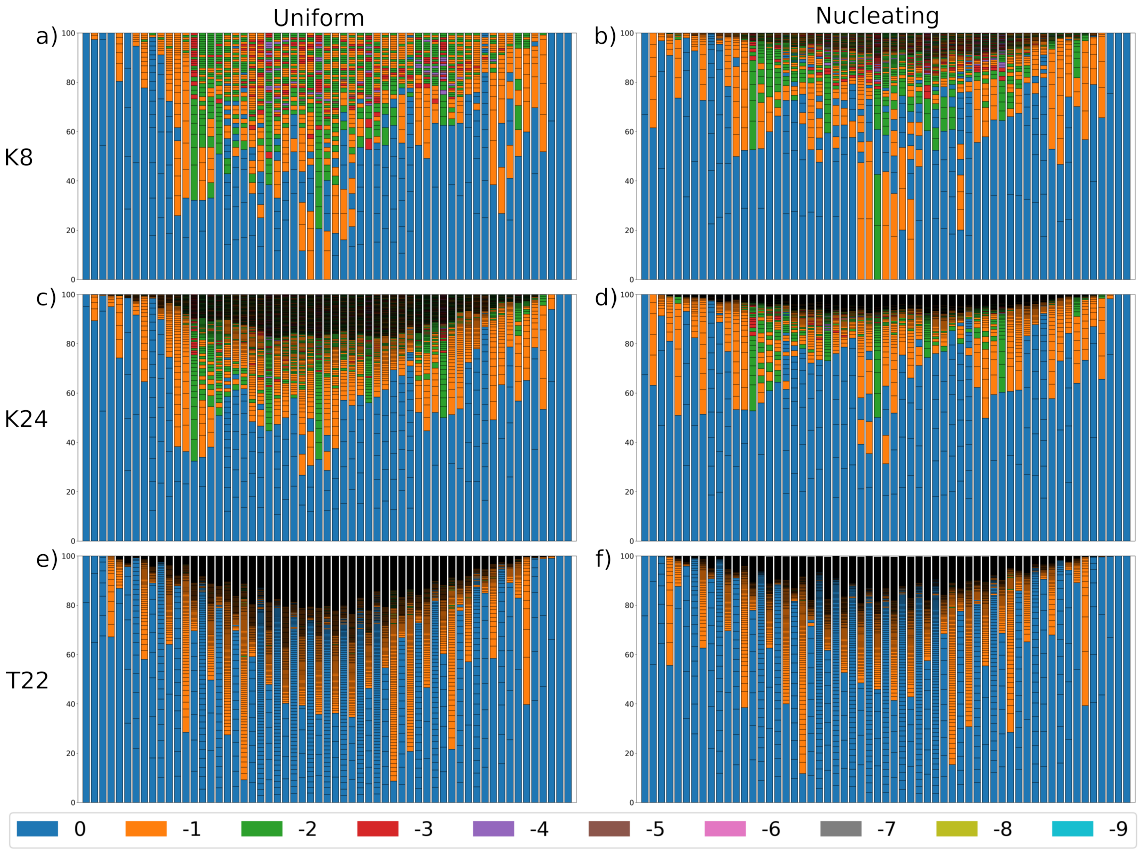

Figure 7.22: Bar chart indicating the frequency that the latest intermediates of a given size occur during the assembly of a capsid using RNA graph (a,b) K8, (c,d) K24 or (e,f) T22. This was done for (a,c,e) uniform RNAs and (b,d,f) nucleating RNAs, with PS distributions shown in Figure 7.19. These graphs were formed using all RNA graphs for both RNAs but these are chosen for display here to illustrate observations. Typically, the nucleating RNAs have a larger proportion of the intermediates in states that have a larger number of capsomer-capsomer interactions. This sample is taken from one simulation of 2000 RNAs. The key, shown at the bottom, represents the difference between the number of interactions formed in a structure and the number formed in the most stable structure of that size.

most precise results. However, in this section the graphs are being considered in a qualitative sense, so as long as the shape is approximately correct, the sample size is large enough. As shown by this test, the sample size will be large enough, even for quite small samples.

Now, six examples of these charts for $T = 3$ viruses are shown in Figure 7.22. The left column are all charts made for uniform RNA and the right column for nucleating RNA. Each row represents a different RNA graph, with the first row being K8, the

second being K24 and the last being T22. Of these, K24 and T22 are isomorphic to each other, yet have vastly different charts here, showing again the effect that the geometry of the capsomers has on the assembly. Comparing K8 and K24 to each other, there are some differences, especially in the heights of the blue columns however the rough shape persists. Some differences are that the chart for K8 seems to show more frequent red and purple bars, with a slightly increased amount of green, suggesting that K24 pushed the assembly along a more stable assembly pathway.

General comparisons can be made upon the effect of adding the nucleation site onto the RNA, shown by the differences between the two columns. There are a lot of cases where the introduction of the nucleation site has increased the size of the blue areas and so overall the stability of the intermediates is improved. However, there are also a few cases (*e.g.* the start/middle sections of K8) where the nucleating site has also increased the size of the orange/green sections of this graph. These nucleating RNAs have generated a higher yield, so whilst this nucleation site leads to more efficient assembly, it does not necessarily lead to all intermediates taking the most stable options during assembly. When simulating STNV in Chapter 6 and removing the nucleation site from the optimised RNA, the assembly of the capsid was dramatically altered, whereas here the changes seem to be less dramatic.

Another interesting observation of these graphs, especially when compared to the example from Figure 7.20 is that the $T = 3$ capsids assemble via intermediates that form fewer capsomer-capsomer interactions than was observed for the dodecahedral capsid. However, the RNA graph which generated the highest yield shown here also has the charts with the fewest green bars, which could still support the conclusion of assembly occurring more efficiently via more stable intermediates.

7.2.5 TREE GRAPHS OF LOWEST ENERGY PATHWAYS

This section describes another method used to analyse the various RNA graphs. This method is a more sophisticated form of the combinatoric procedure discussed in Section 5.1. This method assumes assembly nucleates at one end of the RNA and grows the structure from there. Whenever this new method would add a new capsomer onto the structure, it tests whether adding this capsomer would result in a connected capsomer graph or not, *i.e.* whether it joins the current intermediate structure or not. Instead of looking at all possible intermediates and the exact RNA paths that could assemble, this process finds the three most stable (measured by

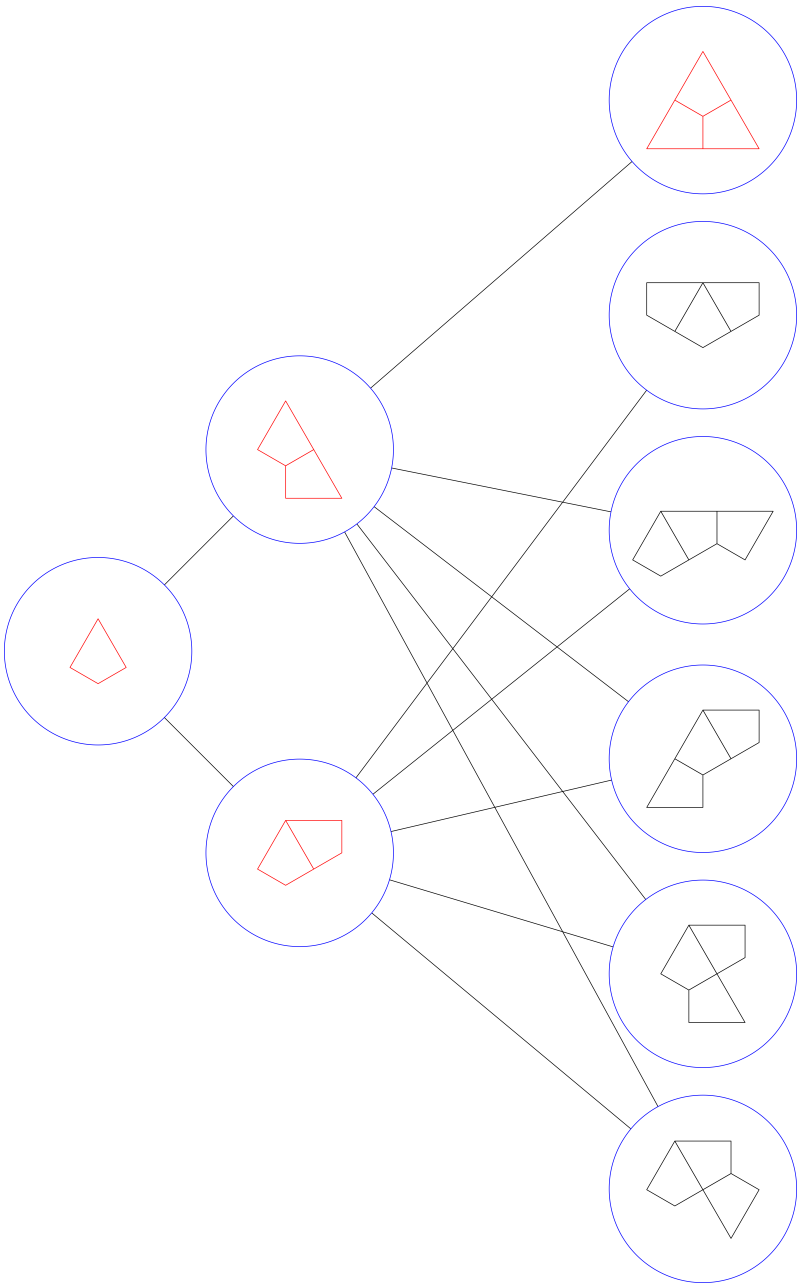

Figure 7.23: A snapshot of a tree diagram as used in this section. Lines between nodes show transitions between nodes due to the addition or removal of a single capsomer. This tree is for a Kite tiling. This example only considers intermediates with up to three capsomers present. The intermediates which form the most interactions are drawn in red, as in the tree diagrams. These diagrams are set up so that they draw only the three intermediates with the most interactions formed. In the third column, there are six different intermediates shown. This is due to a five-way tie. Without foreknowledge of later intermediates, there is no good way to select which of these tied intermediates should be kept and which excluded from further analysis, so all were retained.

the number of capsomer-capsomer interactions in the intermediate) intermediates of a given size. An intermediate here is described only by the capsomers present, rather than considering the RNA layout. Rotationally equivalent intermediates are considered to be equivalent but reflection symmetry is not applied.

When there are ties (meaning that more than 3 intermediates are found which are stable enough to make the top 3, *e.g.* as shown in Figure 7.23), all states that are stable enough are included. As this process assumes that the assembly will nucleate at one end of the RNA and assemble along the RNA from there, it is unable to add in capsomers not bound to a PS, so only Kite and Tri tilings were modelled using this technique. Whilst this process did rely upon the RNA graphs, it did not store all of the paths that could exist for each intermediate. Instead, it only records where the free end of the RNA can sit and where the next PS's capsomer can bind as a result. Additionally, due to how this code runs, it can identify which intermediates can transition to which other intermediates via addition of other capsomers during assembly.

This data was stored in a series of trees, with the first example being Figure 7.23. This shows the intermediates that occur and the possible intermediates that could be formed by adding a capsomer to small a intermediate. Each column lists the intermediates of a given size, with those forming the most capsomer-capsomer interactions drawn in red, to indicate that they are the most stable intermediates. In the following tree diagrams, a larger form of this concept is shown, which includes all sizes of intermediates and does not indicate their shapes. In them, the stability of the intermediates is indicated by a colour scale from red, for the most stable, to black, for the least stable.

Firstly, to give a more digestible example of these trees, the tree for a dodecahedral capsid made up of 12 pentameric capsomers is shown in Figure 7.24. 'it demonstrates that there is a choice of assembly path that features all of the possible most stable intermediates. Also, there are some sub-optimal choices but these can almost always go straight back to the optimal choices by adding another capsomer. The graph shows 39 different intermediates, whereas for the full system there are 73 potential intermediates, demonstrating that when assembly favours the more stable intermediates, then the system is somewhat less complicated than pure combinatorics would suggest. This reduction in available states would likely be even stronger when considering larger capsids, which can have a much larger number of intermediates, thus simplifying the available assembly paths.

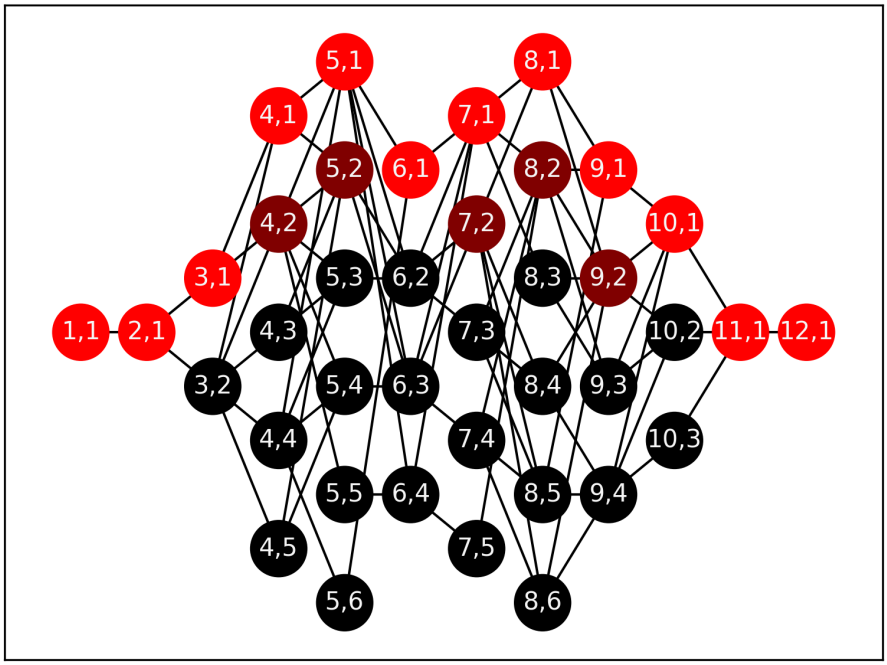

Figure 7.24: Graph of the intermediates with the most capsomer-capsomer interactions during assembly of a dodecahedral capsid formed from 12 pentameric capsomers in the presence of RNA. Limiting the assembly to these intermediates would reduce the number of intermediates in that system from 73 to 39. The nodes are labelled with the number of capsomers present, followed by an identifier. The nodes are colour-coded, where red indicates that it has the largest number of capsomer-capsomer interactions, maroon second most and black means it has, of the intermediates shown, the fewest.

Now, observe Figure 7.25, which shows four tree diagrams for $T = 3$ capsids for a few RNA graphs. Sections (a) and (b) represent RNA graphs T6 and K8, which are isomorphic RNA graphs on different tilings. T6's graph has a flatter and thinner shape than K8's graph, suggesting that the different tilings change the shape of the trees (the same was observed in other isomorphic RNA graphs' trees). The K8 graph had more black columns (where there was a large number of less stable intermediates) than T6, suggesting it presents a larger number of sub-optimal choices.

In (c) and (d), the Figure shows a zoomed in view of the first few sizes of intermediate in the tree. From this, the structures of the tree can be more closely considered. The T6 tree has a larger number of connections than K8, suggesting that there is more flexibility during its assembly, whilst still remaining within stable intermediates. Both show what appear to be dead ends. These are not necessarily dead ends or kinetic traps. They are just states where the assembly cannot add more capsomers without forming intermediates that are insufficiently stable to be present on the tree. This means they are likely less effective pathways but may still be able to assemble.

Next, consider (e) and (f), which represent RNA graphs T22 and K24, respectively. These RNA graphs have a much larger number of edges than T6 and K8, so have a much larger density of different intermediates. As with (a) and (b), these two RNA graphs are isomorphic to each other but also have very different trees here. This is more evidence that the tiling has major effects on assembly, which was observed in many more of these trees that were generated but are not shown here.

Another interesting thing to do is compare the trees within one tiling. To start, (a) and (e), both RNA graphs with Tri tilings, will be compared. In the section that represents smaller intermediates, there is a period where the columns oscillate between there being one most stable state and many less stable states, followed by a smaller number of states but all of them are the most stable options. In the section of the tree representing larger intermediates, the trees are also fairly flat in both cases.

The next pair to consider is (b) and (f), which are both Kite tilings. These two show an even stronger visual similarity than the other two trees. The leftmost section starts gradually increasing in height, followed by a dip and then a sudden increase. Then, the height slowly decreases before a second spike, which forms a plateau with a large number of less stable states. Following the plateau, the height decreases and briefly remains at a low level, with occasional spikes (though these are

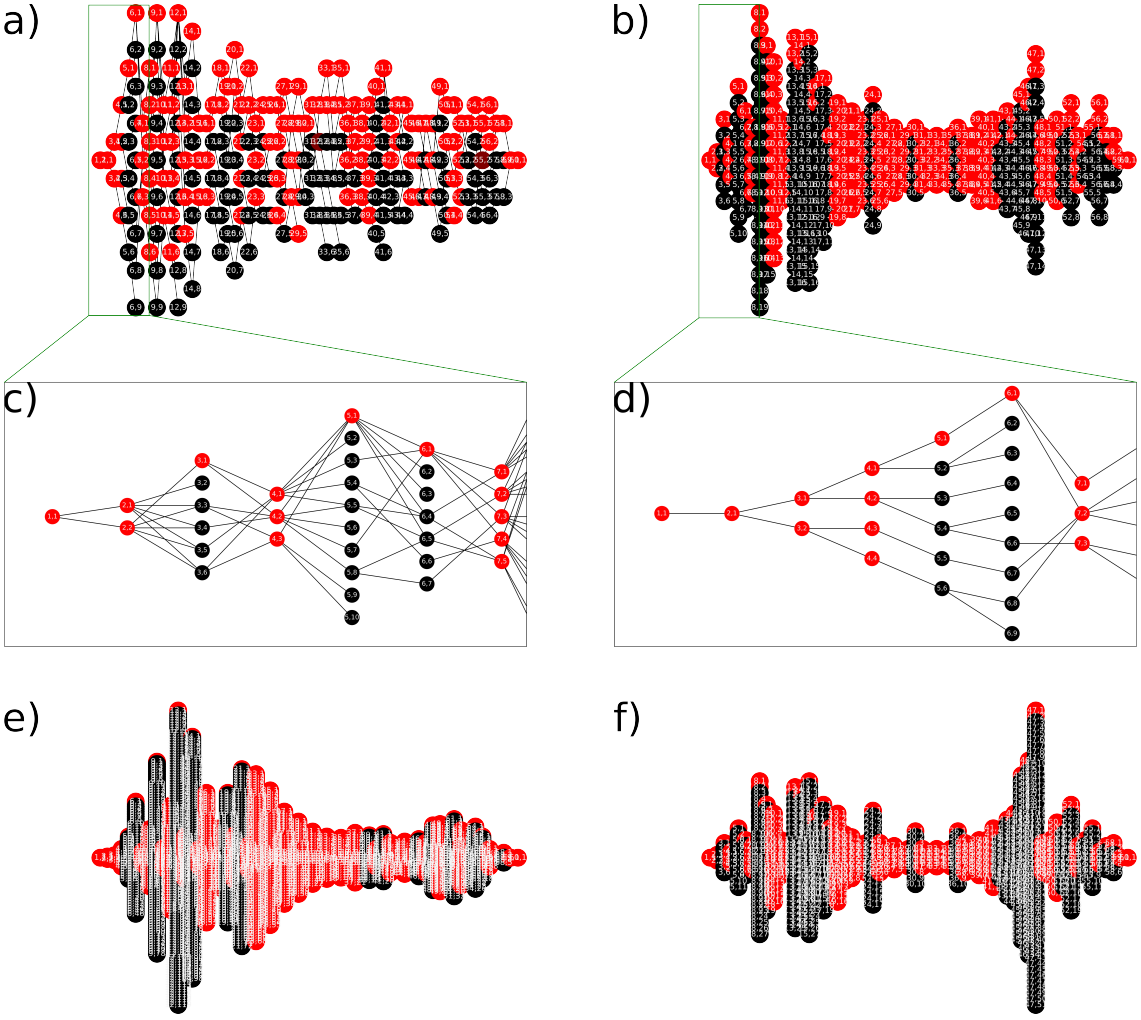

Figure 7.25: A series of trees (similar to Figure 7.24) that feature the intermediates forming the largest number of capsomer-capsomer interactions for a range of RNA graphs and tilings: (a) T6, (b) K8, (e) T22, (f) K24. These RNA graphs are described in Table 7.3. A zoomed in view of (a) and (b) are shown in (c) and (d) respectively. Each column in a graph represents the intermediates with a fixed number of capsomers present. Red nodes have the most capsomer-capsomer interactions and black have the least.

more shallow in (b)). Then, there is a large increase in height before a sudden fall and a few small oscillations to finish the tree.

Within both Kite and Tri tilings, many of these features show up in all of the other trees, though not all of them at all times. The trees for the Kite tilings are much more similar to other Kite tiling's trees than to Tri tilings' trees (and vice versa). Another piece of evidence of how important the tilings are for assembly.

Lastly, the trees in this section can be compared with the bar charts from Section 7.2.4. Figure 7.25e and Figures 7.22e/f all represent the assembly of RNA graph T22. One is using combinatoric analysis and the other is using stochastic simulations. The bar chart shows a lot of spikes where less stable intermediates were used during assembly. Within the tree diagram (though it can be hard to tell without zooming in), these spikes are always within intermediate sizes where there are very few most stable states available, which highlights how well these tree diagrams represent the assembly. On top of that, the bar charts also show evidence that assembly usually only uses the most stable intermediates available (in most cases, intermediates that form have at most one capsomer-capsomer interaction fewer than the most stable intermediates), which supports these trees being of interest.

7.2.6 ΔG_{bond} Fitness Tables In Presence Of RNA

Lastly, there is one more parameter that can be altered, to see the effects of changing it upon the assembly. This parameter is the strength of interactions between capsomers, ΔG_{bond}. Changes to this parameter were covered for the RNA-free case in Section 7.1.2 and now this will be altered whilst the RNA is present. This will be applied to the RNA graphs which had their RNA optimised, *i.e.* K14, R14 and T12. This helps to indicate how well this optimisation works and whether other improvements are available.

7.2.6.1 Effects Of Changing Uniform ΔG_{bond}

Initially, the values of ΔG_{bond} is assumed to be constant across all capsomer-capsomer interactions. This allows a simpler view of the parameter space whilst also altering a few other parameters, to give an idea of the areas of interest.

Initially, a range of different values of ΔG_{bond} and a range of different RNAs were investigated, across all three RNA graphs. For each data point, one simulation of

2000 RNAs was implemented and the number of assembled capsids at the end was recorded, as shown in Figure 7.26.

There is a range of important observations here. The first is that if the optimisation had been solely to generate the highest possible assembly yield (rather than trying to indicate differences of assembly across tilings), then the values of ΔG_{bond} could have been altered and the assemblies would have generated a higher yield. The Kite and Tri tilings would have preferred a lower ΔG_{bond} but the Rhomb tiling would have generated a greater yield with either a higher or a lower value of ΔG_{bond}.

The RNAs in the Kite and Tri assemblies can easily be categorised into two distinct groups: the high yield and the low yield (and the unassembled, which only includes 12Uni). These groups also correlate with the presence and absence of a nucleating site at one end of the RNA. Note that RNA 8Nuc drops in efficiency when it has sufficiently low ΔG_{bond}. However, it typically occupies the efficiency of the more effective group.

Another noteworthy RNA is 4Nuc, which has a nucleation site, (*i.e.* 5 strongly binding PSs) and then all other PSs are weakly binding. In Rhomb, this is significantly more effective than any other RNA explored and in Tri it is still more effective than the alternatives. In the Kite, it is usually the most effective (competing with AvgNuc). However, for the value of ΔG_{bond} where the RNA was optimised, the most effective RNA was the optimised RNA, though not by much. This suggests that the parameter space of PS distributions was not sufficiently sampled to obtain the best available RNA, as for two RNA graphs, a better RNA was obtained arbitrarily.

From the graphs, it can be concluded that the K14 is the most dependent on ΔG_{bond} of the three RNA graphs, as it has the largest peak of the three. It can also be observed that graph R14 is the most dependent on its PS distribution, with the value of ΔG_{bond} being less significant here. Additionally, T12 is not affected as strongly by RNA choice or ΔG_{bond}, as the graphs are fairly flat and a range of different RNAs resulted in similar yields.

7.2.6.2 Effects Of Changing Non-Uniform ΔG_{bond}

Section 7.1.2 introduced the concept of different strengths of ΔG_{bond} for different capsomer-capsomer boundaries and explored this for assemblies in the absence of RNA. Each capsomer for the three $T = 3$ tilings can form at least two different capsomer-capsomer interactions. This is developed here for the case where RNA is present. As before, Bond A is the capsomer-capsomer interaction about the 5-fold

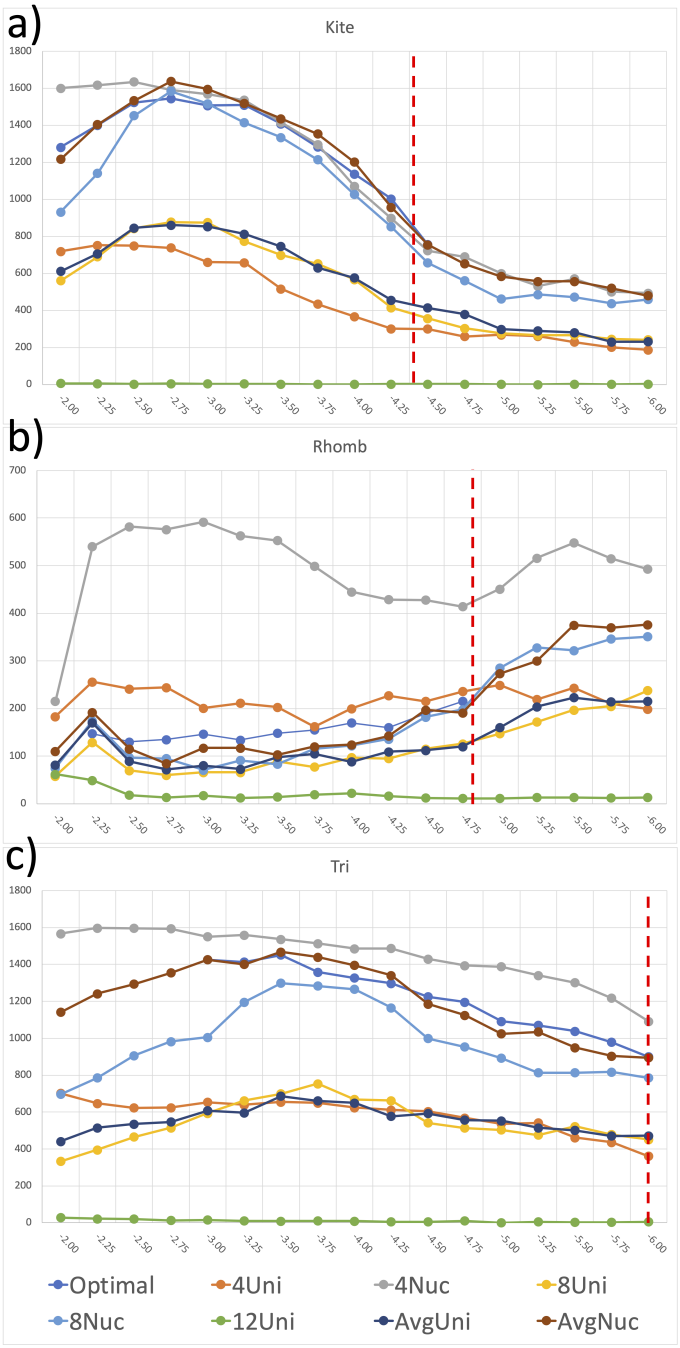

Figure 7.26: Graphs indicating the number of assembled capsids for RNA graphs (a) K14, (b) R14 and (c) T12, when assembling with a variety of different RNAs and at a range of values of ΔG_{bond}. The ΔG_{bond} used during the optimisation is given as a red dashed line, as this was the value determined in Section 7.1 to give an even playing field. The optimised RNA is the one that was found using the optimisation procedure given above. The "XUni" RNAs have a uniform $\Delta G_{rna} = X$ kcal/mol. The "XNuc" RNAs are the same as the "XUni" RNAs with a nucleation site of 5 PSs with $\Delta G_{rna} = 12$ kcal/mol at one end of the RNA. For "Avg X" RNAs the values of ΔG_{rna} is set to the average value of all ΔG_{rna}s across the optimised RNA. The data for ΔG_{bond} for the optimised RNAs, is shown for (a) up to 4.5 kcal/mol, (b) between 2.25 and 4.75 kcal/mol or (c) from 3.0 kcal/mol.

axis and Bond B is the other capsomer-capsomer interaction that forms. Within this section, all of the Rhomb tiling's interactions that involve the capsomer that sits across the 2-fold axis are assumed to be equivalent.

Figure 7.27 shows three heatmaps from simulations where the strength of these interactions was varied. Each of these heatmaps shows a skew, which favours one capsomer-capsomer interface over the other. The Kite has a greater yield when Bond B is stronger, *i.e.* interactions about the 3-fold axis, which are thus more significant to the assembly dynamics. The Tri has a slight preference for Bond B, which stretches over the 2-fold axis, though it still needs a certain minimal strength for Bond A. This could be due to that contact being the only option to build a capsid after a 5-fold axis is complete. The Rhomb is extremely skewed to favour Bond B, still assembling with comparably high yield to the rest of the heatmap when Bond A is at the lowest value tested. This is likely due to two factors. The first is that there are twice as many Bond B's in the Rhomb capsid than Bond A's. The other reason is that Bond B always includes one of the capsomers that sit over the 2-fold axis, which do not bind to the RNA. These capsomers will therefore not be stabilised by the RNA, whereas the other capsomers rely on a mix of A and B, in addition to the stabilisation due to RNA contacts.

Figure 7.27 can be compared with the RNA-free case in Figure 7.6. The first observation is that the shape of the heatmap has changed, which is due to the presence of the RNA and hence reflects the choice of RNA graph. Different choices of RNA graphs would likely result in differently shaped heatmaps. Even though Figure 7.27 shows a smaller range of ΔG_{bond} values, it still appears to contain most of the plateaus, suggesting that effective assembly around the RNA will occur for a narrower range of parameter values. Another important observation is that assembly still occurs for much smaller values of ΔG^{A}_{bond} and ΔG^{B}_{bond}, even if both are small. Using the range of values of ΔG^{A}_{bond} and ΔG^{B}_{bond} shown, both the Kite and Tri tiling always assemble some completed capsids. This demonstrates the stabilising effect of the RNA upon the assembly.

7.2.6.3 Non-Uniform Variation Of All Three ΔG_{bond} Values In The Rhomb Tiling

Lastly, as covered in Section 7.1.3 and Figure 7.5, the Rhomb tiling actually has three distinct capsomer interactions, not just two. In this section, the effect of changing the strength of these interactions is covered. Figure 7.28 shows the relative positions

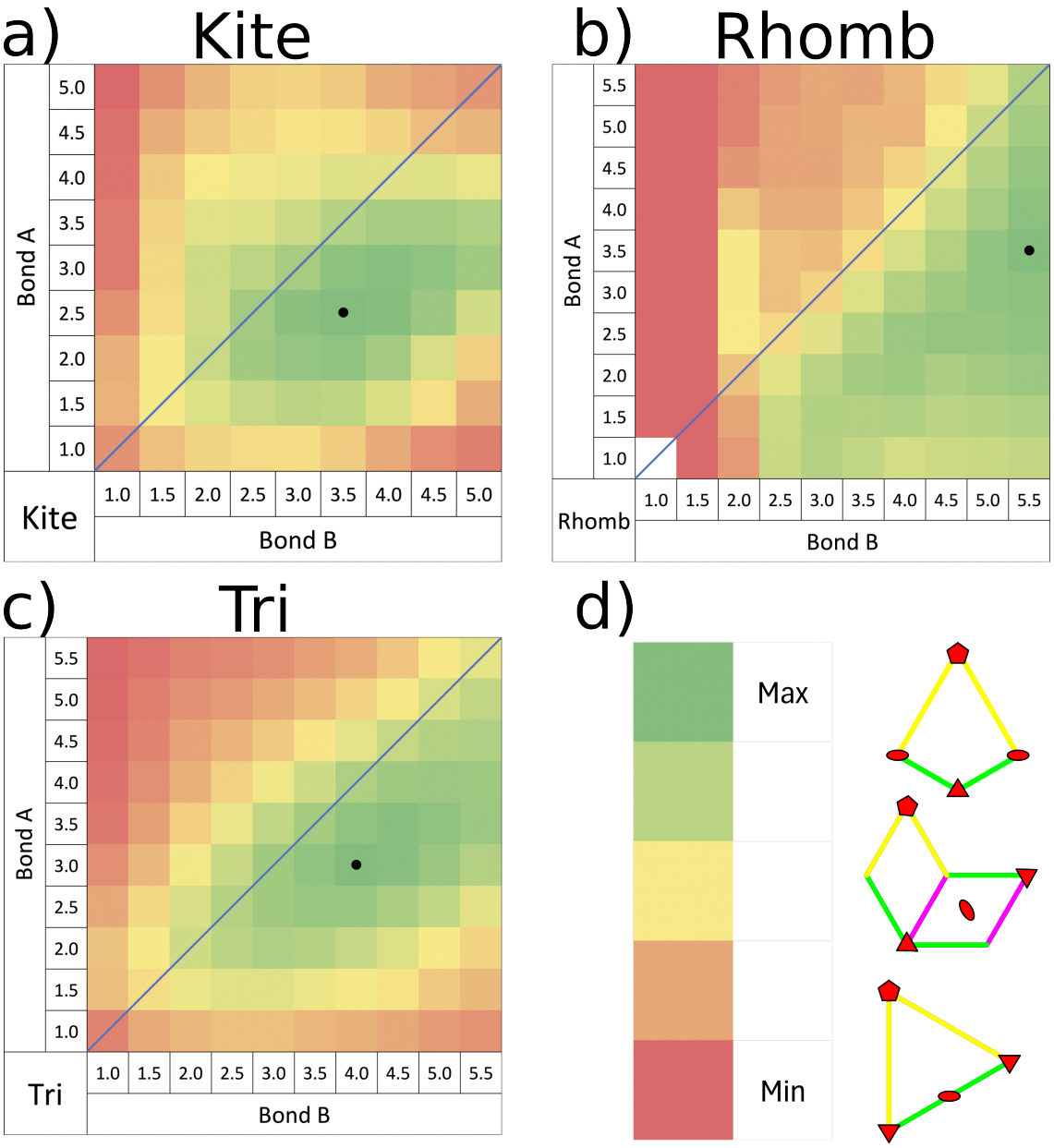

Figure 7.27: Heatmaps showing the assembly efficiencies of the optimised RNAs when different values of ΔG^A_{bond} and ΔG^B_{bond} are used. They contain the results for (a) Kite, (b) Rhomb and (c) Tri tilings. There is a scale for all of the heatmaps in (d), alongside a diagrammatic representation of the bonds in each tiling. Yellow indicates Bond A, Green indicates Bond B and Pink indicates Bond C, which was set as equal to Bond B here. Graphs are based on the average over three simulations of 2000 RNAs. The range of the heatmap for Kite is $[337, 1785]$, for the Rhomb $[0, 452]$ and for the Tri $[162, 1634]$.

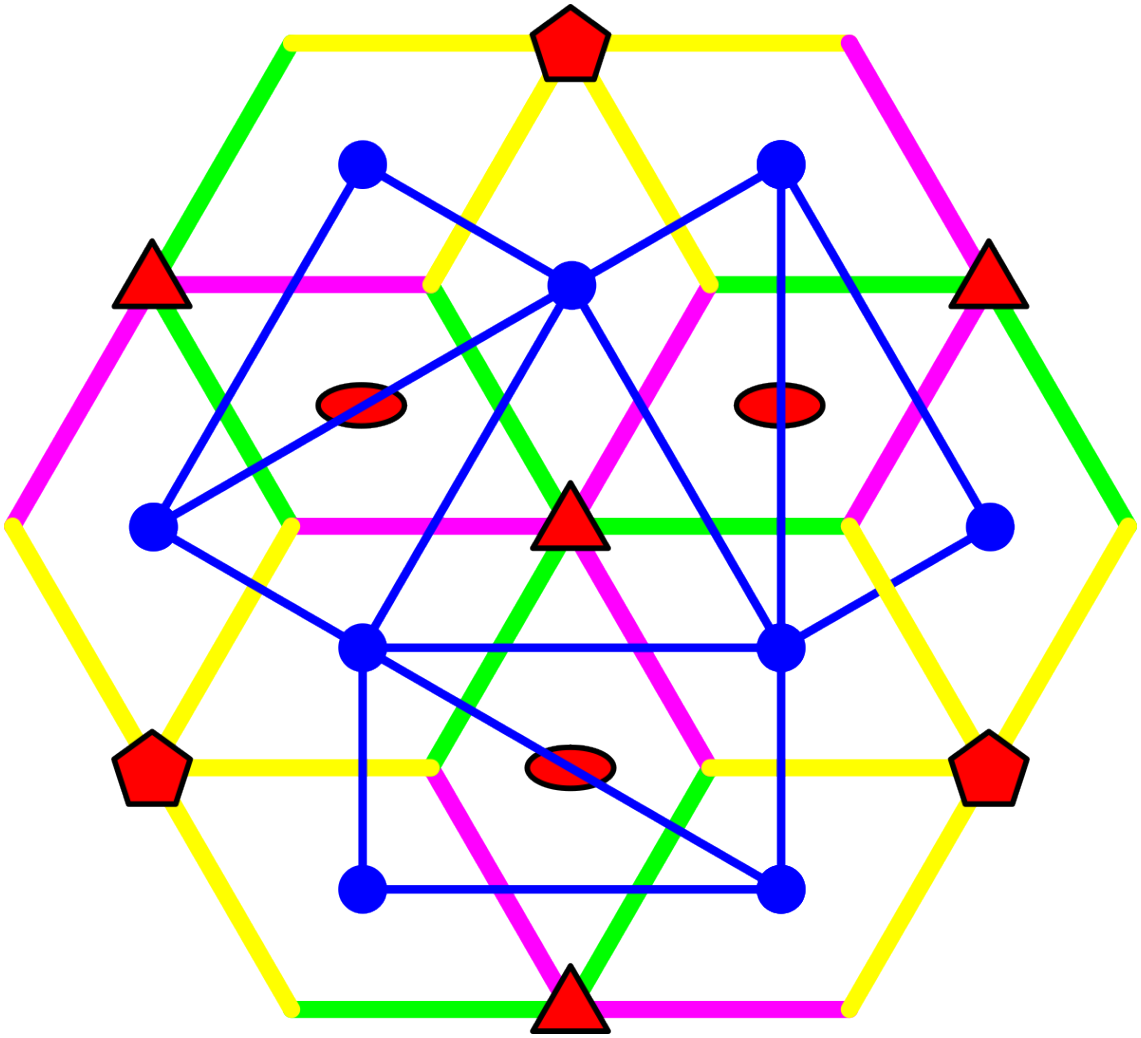

Figure 7.28: Figure illustrating how the three different capsomer-capsomer interfaces sit within the RNA graph. A small patch of the Rhombic tiling is shown, where the three different capsomer interfaces are superimposed in different colours. Yellow edges represent Bond A, green edges represent Bond B and pink edges represent Bond C. On top of this, the RNA graph for the R14 tiling is shown as blue lines. The pentagons, triangles and ovals represent the 5-, 3- and 2-fold symmetry axes of the capsid.

of the three different boundaries. This shows that for the R14 RNA graph, Bond B is important to enable the RNA to use edge δ, which reaches across the capsomers that sit on the 2-fold axis. This means that in the heatmaps there is likely to be a skew, which was not present during the heatmaps in Section 7.1.3 which shows another effect of the RNA on the assembly.

Figure 7.29 shows heatmaps for the assembly of the Rhombic tiling, where each

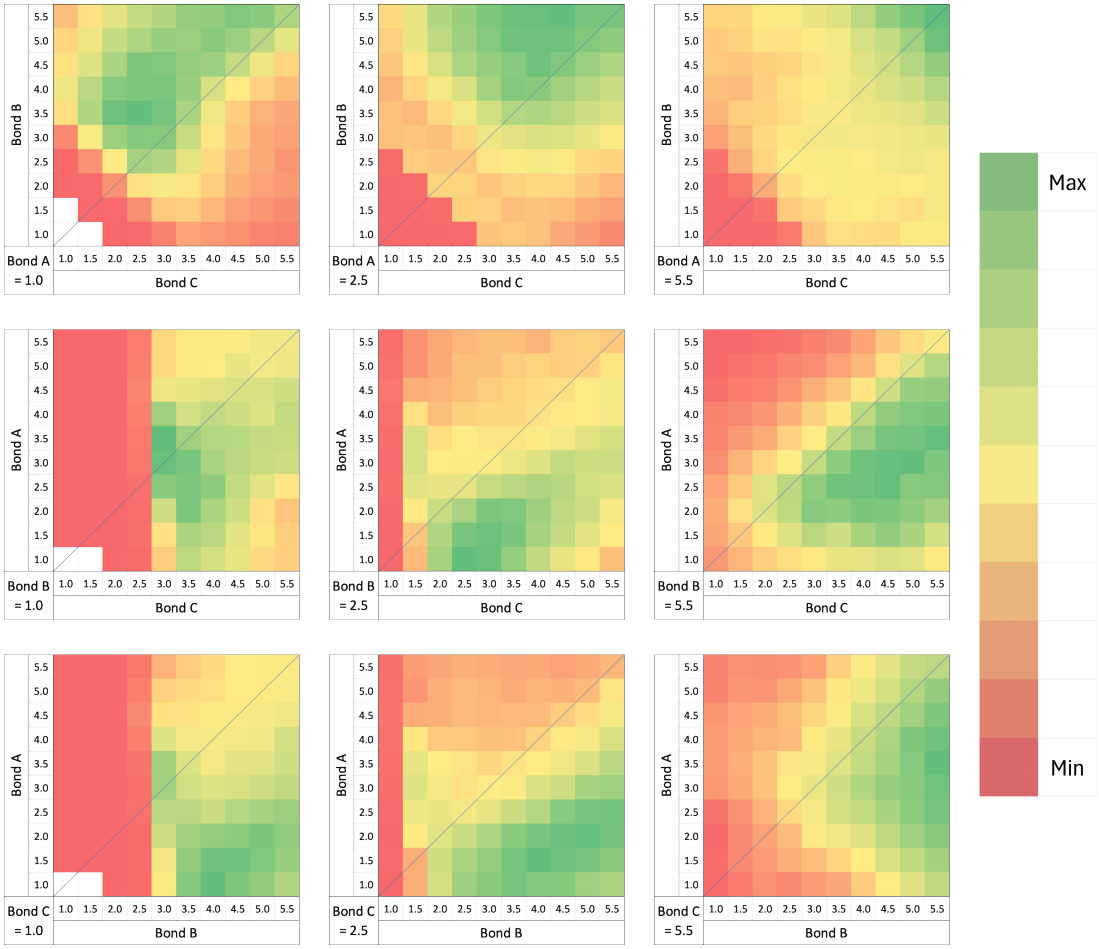

Figure 7.29: Heatmaps showing the assembly efficiencies of the optimised RNA for RNA graph R14, when different values of ΔG^A_{bond}, ΔG^B_{bond} and ΔG^C_{bond} are used. They were generated using the average result from three simulations of 2000 RNAs for each square. White squares indicate that the simulations took too long to complete. Typically the less effective the assembly, the slower the simulations ran. So, observing where these white squares sit suggests these would all have very low yields. These heatmaps all have independent scales where green has the largest number of capsids formed and red is the smallest number of capsids formed. The largest and smallest values for each heatmap are recorded in Table 7.6.

of the three capsomer-capsomer interaction strengths was altered independently.

Initially, consider the first row, where ΔG^A_{bond} is fixed within each heatmap and increases from left to right. When ΔG^A_{bond} is small, there is a clear skew where a larger value of ΔG^B_{bond} leads to a higher number of assembled capsids. As ΔG^A_{bond} increases, this skew reduces until the final heatmap, where is it almost symmetric. This is different to Figure 7.9, where the equivalent data is shown for the RNA free

Fixed Bond	Fixed ΔG_{bond}					
	1.0		2.5		5.5	
	Max	Min	Max	Min	Max	Min
A	351	0	451	0	284	0
B	149	0	284	1	453	49
C	217	0	384	2	452	35

Table 7.6: Figure 7.28 has nine heatmaps. This table contains the maximal and minimal numerical values for capsids assembled in each scenario.

version and is symmetric. It makes sense that Bond B is the favoured interface as when a capsomer joins at this interface then another edge is available for the RNA, to go across, as shown in Figure 7.28 with a green edge.

Looking at the other two rows, there is a strong skew, which suggests that Bonds B and C are both more important than Bond A, as the efficiency below the line is higher than above in all cases. This could be due to Bond A being present only on capsomers which bind to the RNA and the RNA will cause an additional stabilising effect.

Another conclusion that can be made is that there appears to be a strict minimum value for ΔG^B_{bond} or ΔG^C_{bond}. Observing the lower two heatmaps show that if both are small, regardless of how strong ΔG^A_{bond} is, the capsid will not assemble. Even if one is increased, if the other is still low then it will still not assemble, as shown in the second column. Comparing the lower two heatmaps in the third column, B is again favoured, where capsid structures with high ΔG^B_{bond} and low ΔG^C_{bond} still assembling but the inverse being less effective. Lastly, the first row also has a diagonal line where if ΔG^B_{bond} and ΔG^C_{bond} are both low, then it is incapable of assembling.

7.3 CONCLUSION

In conclusion, there are a large number of variables that will affect the efficiency of the assembly of viral capsids. Some give smaller effects, for example whether the capsomers are ramped in or the simulation begins with them already present, which can affect the efficiency for larger values of ΔG_{bond} but is minor otherwise. Others are more significant, such as choosing a tiling, as this determines the shape of the capsomer, which in turn determines the adjacencies of the capsomer graph. And changing the capsomer graph was shown to have significant impact on the assembly

of the capsids, whether RNA was present or absent. Additionally, changing the values of ΔG^A_{bond}, ΔG^B_{bond} and ΔG^C_{bond} (when appropriate) was also an impactful change that affected the capsid's assembly. Each tiling typically had one bond it favoured over the others and making this bond bind significantly more strongly than the other would lead to more efficient assembly, whether the RNA was present or not.

The RNA's presence was also an important factor, as it allowed efficient assembly with smaller values of ΔG_{bond}. However, considering the RNA leads to more choices, one being which RNA graphs to use. For each tiling presented, there are over 20 different valid RNA graphs and it is challenging to determine which best reflects the actual biology. Another set of parameters that can be changed are the values of ΔG_{rna} along the length of the RNA. In this chapter, there have been 60 PSs along the RNA, so there are 60 different values for this parameter that can be altered. However, the exact values of each ΔG_{rna} are not as important as the shape of the distribution. For example, having a nucleation site is much more effective than having a uniform distribution. The RNA also affects how changing ΔG^A_{bond}, ΔG^B_{bond} and ΔG^C_{bond} (when appropriate) impacts assembly. In the Rhomb assemblies, adding RNA with some RNA graphs will introduce an asymmetry between Bonds B and C that was not observed when the RNA was not present.

The paths that the RNA traced out within $T = 3$ capsids was not observed to have any similarities across capsids, so the intermediates formed during assembly were investigated. This showed that most of the time, the assembly favoured the more stable intermediate structures, with the vast majority of intermediates analysed only having either the maximal possible number of capsomer-capsomer interactions or only one fewer. Using sophisticated combinatorics, there was shown to still be a large number of intermediate structures that satisfied this condition.

— 8 —

Examining Disassembly Using Stochastic Simulations

So far, the focus has mainly been on the assembly of viruses. Assembly is an important process during the replication cycle of viruses but it is not the only one. The basic steps of the replication cycle shared by most viruses can broadly be summarised as follows:

1. Virus enters host cell
2. Virus disassembles to release genetic information into the host cell
3. Host cell replicates virus' genetic information and produces coat proteins
4. Coat proteins assemble into a completed capsid containing genetic information
5. Host cell releases assembled capsids to the environment
6. Virus finds new host cell and the cycle begins anew.

This chapter will focus on the disassembly step of this cycle. This is the process by which fully assembled capsids can release their genetic information into a cell, by breaking apart. It is an important part of the viral replication cycle and can also have applications within nanotechnology, as virus-like particles have potential usage in drug delivery. In addition, disassembly can be modelled using similar techniques to assembly and this is a natural extension of earlier chapters. The techniques developed during this thesis which model viral assembly are repurposed during this chapter, to model disassembly instead.

8.1 PRIOR WORK ON DISASSEMBLY

Three papers shall be briefly discussed and used to compare with the work presented in this thesis.

The first paper is *Observed Hysteresis of Virus Capsid Disassembly Is Implicit in Kinetic Models of Assembly* by Sushmita Singh and Adam Zlotnick [48]. In this paper, they consider HBV, a $T = 4$ virus composed of 120 dimers. It featured both experimental results and computational models. The computational models were done by assuming that only the most stable intermediates form, defining rate equations and then using numerical integration, similar to other work by Zlotnick [59].

In this work, they made a number of observations. The first was that is took a long time for the system to reach equilibrium, longer than in assembly simulations. Another was that they found that experimentally there was a higher presence of intermediates during disassembly experiments than was seen in their assembly data. They found hysteresis would mask changes in contact energies (caused by altering the salt concentration in their experimental system) between capsomers, in all but extreme cases. One suggestion was that viral capsids may need a triggering event to initialise the disassembly process.

The next paper is *How to Disassemble a Virus Capsid, a Computational Approach* by Claudio Alexandre Piedade, António E. N. Ferreira and Carlos Cordeiro [46]. They focus on a range of $T = 1$ viruses. For each virus, they used the atomic coordinates of a structure and formed a capsomer graph, where each edge indicated that the two proteins are able to interact. Then, they counted the number of bonds that each protein made to each of its neighbours. This was based on contributions from hydrogen bonds, salt bridges and hydrophobic contacts. During their analysis, they used different weightings for each of these bond types.

To model the disassembly, they allowed one protein per reaction to leave the structure. If a removal meant the capsid was cleaved into two, then the larger intermediate was kept. They formed a tree of disassembly paths that allowed them to find the most probable intermediates, by assuming the most stable intermediate would be the one to form.

This allowed them to find plausible pathways for the removal of up to 5 proteins. One conclusion from this is that many viruses (especially closely related viruses) are likely to begin disassembly with highly similar paths, for example most viruses lose

capsomers in a trimer interaction first.

The last paper to be presented is *Percolation Theory Reveals Biophysical Properties of Virus-like Particles* by Nicholas E. Brunk and Reidun Twarock [5]. This begins with the statement that for $T = 3$ viruses, there are only three topologically distinct tilings made up of identical capsomers: kite, triangular and rhombic tiles.

For each tiling, an interaction network, *i.e.* a graph containing the capsomer interactions, was generated and then the graph underwent tests. The first test was to randomly remove one node at a time until the capsid cleaved into two disconnected graphs and counted how many nodes had been removed. The number of capsomers out of the total removed when the capsid dissociated into two disjoint parts was used as a metric for when the capsid had fallen apart. This test was repeated 10,000 times for each graph to ensure the results were representative and the number of nodes removed was stored as a fraction of the number of nodes removed divided by the total number of nodes in the original graph, to allow comparison between graphs with differing numbers of nodes. The probability that the graph was still connected with a given fraction of nodes remaining was calculated.

The other test performed was to remove edges from the graph, rather than nodes. Apart from this, the rest of the test was the same. The fraction of edges removed was stored and was used to give the probability that a graph was still connected after a given fraction of its edges had been removed.

The result of both these calculations were in agreement, demonstrating that the interaction network is sufficient to capture essential features of capsid disassembly. Whether nodes or edges were removed, the most stable tiling was the Kite tiling, followed by the Rhomb tiling and then the Tri tiling. This was continued for Rhomb and Tri tilings for larger T-Numbers, as Kite tilings have only been observed for $T = 3$ tilings. The trend was that Tri tilings continued to be less stable than Rhomb tilings. Additionally, as the T-Number increased, the overall stability of the tilings decreased.

8.2 METHODS USED TO MODEL DISASSEMBLY

Assembly and disassembly are both processes that are based upon the same reactions, where at any point, a capsomer may join or leave a capsid's current intermediate structure. Thus, with only small adjustments to the original assembly code, it can

be adapted to model disassembly, too. This is done by setting the initial state of the system to be a given number of complete capsids in the absence of any free capsomers. One limitation to this is that the internal arrangement of the RNA is unknown, so highly detailed knowledge regarding where the RNA sits on the interior of a completed capsid is needed to model capsid disassembly in the presence of RNA. As a result, only RNA-free disassembly is modelled in this chapter. Then, once the initial state is set up, simulations are run in the same way as for the assembly simulations detailed in Chapter 4, by giving the simulation information on the capsid's capsomer graph and interaction strengths, then sampling reactions.

8.3 RESULTS OF DISASSEMBLY SIMULATIONS

A series of both assembly and disassembly simulations were implemented, using a range of different tile types to represent capsomers, with different values for capsomer-capsomer interaction energies (ΔG_{bond}). Effects of different time limits (10s, 100s, 1000s and 10,000s) on the simulations was also explored, in order to investigate how timeframes impact the ability to reach equilibrium. In the assembly simulations, capsomers were ramped in, emulating *in vivo* conditions, to avoid the yield dropping off at higher values of ΔG_{bond}, as shown in Section 7.1.1.

The results of this are shown in Figure 8.1, which shows the number of fully formed capsids after both assembly and disassembly simulations were run, across multiple timescales, using each tiling and with a range of values of ΔG_{bond} from 2.0 kcal/mol to 5.0 kcal/mol in steps of 0.2 kcal/mol. The low yield after shorter simulations show that full assembly requires significantly longer time than they took place over. One caveat is that this could be due to the ramp not having produced enough capsomers, which has lowered the yield but it is not the sole reason. In the 100s simulations, there will be approximately enough capsomers generated to form 667 (33%) capsids in the Kite/Tri or 444 (22%) in the Rhomb, but the 100s simulations never generated this many capsids, even if their 1000s simulations assemble the maximal number of 2000 capsids.

The disassembly simulations do not show such a stark difference in number of capsids remaining. One reason for this could be due to the measurement criteria, where in both cases a capsid is considered complete (either it is assembled or it is not disassembled) if it has all its capsids and incomplete (either it is not assembled or it

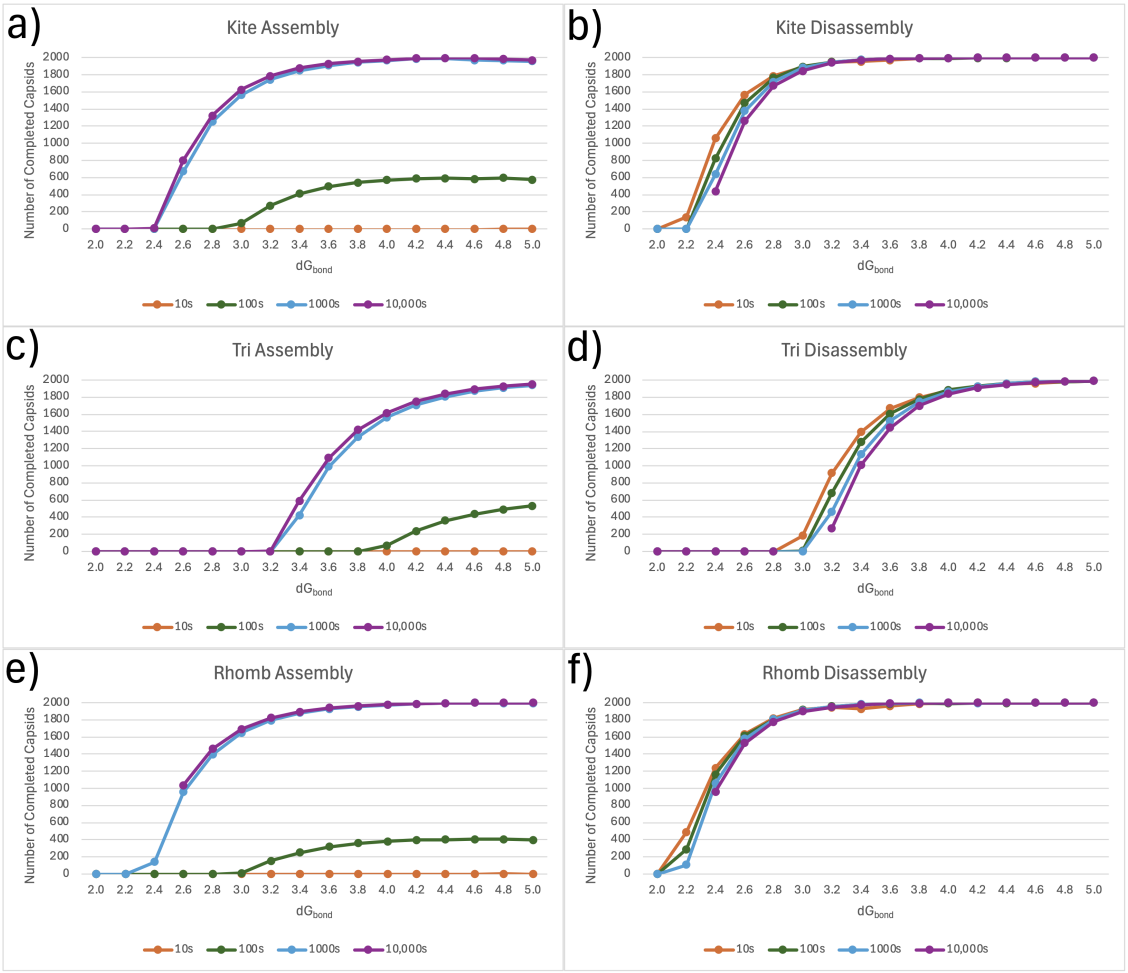

Figure 8.1: Graphs containing the yield of (a,c,e) assembly simulations and (b,d,f) disassembly simulations in (a,b) Kite, (c,d) Tri and (e,f) Rhomb tilings for a range of values of ΔG_{bond}. The assembly simulations had enough capsomers to generate up to 2000 completed capsids. However, the capsomers were ramped in, so they were not all present until (a,c) 300s or (e) 450s into the simulation. The disassembly simulations began with 2000 completed capsids. Due to the extremely long simulation times, some data for the simulations over 10,000s (usually at lower ΔG_{bond} values) is missing.

is disassembled). This means that when a capsid undergoes a reaction where it loses a capsomer, it is no longer a complete capsid. However, a free capsomer needs to go through 60 reactions to become a complete capsid, so it makes sense that it needs more time for assembly than disassembly. In addition, there is the potential for kinetic traps during the assembly simulations, which is not an issue in disassembly simulations. Piedade *et al*'s work showed different shapes of capsomers with different interaction strengths can have clear differences in their disassembly [46]. This can be seen as the tilings all give different assembly and disassembly efficiencies for different values of ΔG_{bond} and different time spans.

The graphs all take a sigmoidal shape, so they have a stable plateau for both low and high ΔG_{bond} values and a relatively steep curve between them. The position of these curves differ by tiling, as expected from work by Brunk and Twarock [5] and change depending on whether it is an assembly or disassembly simulation. However, regardless of whether it is an assembly or disassembly simulation, the relative positions of these curves are consistent. Rhomb and Kite tilings have their curves very close together and were both able to assemble with much smaller ΔG_{bond} values than the Tri tiling.

Now, considering Figure 8.2, the hysteresis of these simulations is shown. As the simulations are given a longer time limit, the number of capsids at the end becomes more similar, indicating that they may have approached a state closer to equilibrium. This is shown to take a long time, as letting the simulation run for 10 times the initial time frame still does not cause the two lines to meet. As mentioned earlier, this could also be due to kinetic traps that can prevent some of the assembling capsids from ever fully assembling. However, the graphs do show a clear sign of hysteresis, which was predicted by Singh and Zlotnick [48]. They also predicted that a trigger of some kind is likely to assist with disassembly. This would make sense with these observations, as the initial reaction needed to start disassembly in this model is also the slowest as it would need to break three (Tri) or four (Kite/Rhomb) capsomer-capsomer interactions. If a trigger reduced the strength of these interactions, then the behaviour of the capsid would transition to the other side of the sigmoidal curve and have much reduced stabilities.

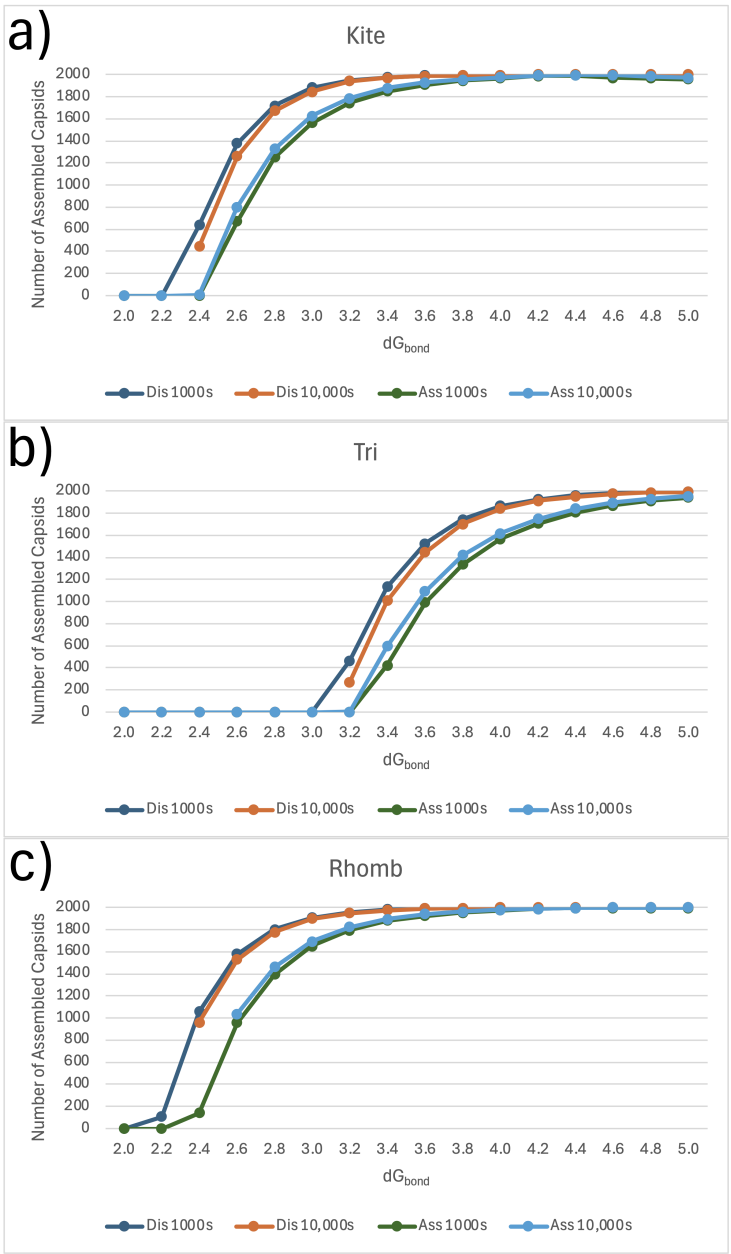

Figure 8.2: Superimposition of the data from Figure 8.1 in a more concise way. This reveals the hysteresis of assembly and disassembly, as there are far more completed capsids in the disassembly simulations than in the assembly simulations. As the simulation time is increased, the two lines move closer to each other, albeit slowly.

— 9 —

Discussion

In this thesis, different stochastic methods and approaches to modelling virus assembly and disassembly were presented. Here, some of the observations and results from this text are briefly summarised.

Chapter 2 covered both ODE and Stochastic models, alongside some virus biology and mathematical models of virus architecture. Through this, the choice of Stochastic Modelling, specifically Gillespie algorithms, were given as the preferred choice of technique to model virus assembly, due to them treating each capsomer and capsid as discrete elements.

However, before going into depth with stochastic models, two examples of ODE models were presented in Chapter 3. These models both considered viruses made up of twelve pentameric capsomers. The first model showed the stabilising effect of including the RNA in viral assembly via an equilibrium model. Including the RNA lowered the necessary strength of capsomer-capsomer interactions for assembly to occur and identified the minimal necessary strength of these interactions needed for assembly. This developed on the work of Zlotnik, who applied the equilibrium model to the RNA-free case [59]. The second model showed that the value of ΔG_{bond} in this system determines whether the capsomers form completed capsids or remain unassembled at equilibrium, though it may take a long time to reach this equilibrium state.

Then the computational techniques used to produce a stochastic simulation of viral assembly is described in Chapter 4, followed by Chapter 5, where methods to analyse the results of the stochastic simulations are described. These techniques were used within the three following chapters. These chapters improve upon previous work, which included a similar model that worked on a smaller range of viruses [19].

Chapter 6 covers models of assembly for STNV, a $T = 1$ virus. In this chapter, an RNA graph and a PS distribution were identified that lead to $> 95\%$ assembly efficiency. Assembly using this RNA graph and PS distribution combination produced a large amount of repetition of the RNA paths formed within the capsid, which was partially due to the limited number of available paths. It was also shown that every edge that was included or excluded in the RNA graph is important, as adding an extra edge to the most efficient RNA graph leads to a much reduced yield. Also, RNA graphs with a larger vertex degree did not necessarily have a higher yield. STNV is well studied by experimental techniques [34, 35, 42, 43, 6, 22] but has limited papers investigating it computationally.

Chapter 7 covered the assembly of each of the $T = 3$ tilings. This chapter showed how changing the geometry of both the RNA graph and the capsomer graph can affect the assembly of the capsid. It showed that different tilings need different values of ΔG_{bond} to assemble effectively and that letting different capsomer-capsomer boundaries have different ΔG_{bond} values also increases assembly efficiency. It helped develop the idea that assembly strongly favours the most stable intermediates available. In addition, it validates the prior observation that assembly efficiency is increased by having a nucleation site on the RNA. It also showed some limitations of the model. When the ΔG_{bond} is too high, this model does not consider defects, which would occur in the biology. Also, with a larger virus comes more PSs, which leads to a larger space of paths that the RNA can take within the capsid. With more options for paths, the assembly was less likely to specialise into a single, highly effective path, which was necessary for STNV to obtain a high assembly efficiency. In cells, larger viruses are able to assemble effectively but this model did not demonstrate this. Pre-existing research of T=3 virus assembly using stochastic state based methods typically modelled them in the absence of RNA and allowed far fewer capsids to form [49].

In both of the previous two chapters, it was shown that the PS distributions of the RNA is significant. The specific binding affinity values of each PS are not as important as the presence of a series of adjacent, strongly binding sites, which the structure nucleates from. Ideally these binding sites are positioned at one end of the RNA. Most of the other sites should be more weakly binding, to help avoid kinetic traps.

Lastly, Chapter 8 covers the dynamics of RNA-free disassembly, to compare it to RNA-free assembly. When comparing assembly and disassembly, hysteresis was

observed, likely for a range of different reasons. So, two systems using the same parameter values but different starting points will take a long time to reach the same state. This develops on previous work where this was demonstrated using equilibrium based models, by showing it is also true in stochastic models [48].

9.1 LIMITATIONS AND POTENTIAL FUTURE WORK

Now, the text will outline a few limitations of this work and suggestions of how it could be further developed, to more accurately model assembly and disassembly dynamics. Some are concepts that were left unexplored and others include directions in which the simulations could be improved.

When considering dodecahedral capsids made of 12 pentameric capsomers, there are limited examples of viruses that form from so few capsomers. The formation of the capsomers from individual proteins ideally would not be dismissed too. In the pseudo-equilibrium formulation, a volume containing only 12 capsomers was modelled. Viruses will often form thousands of complete capsids in each cell, which means that there will be tens of thousands of capsomers during assembly [12]. As a result, it would be rare to have a volume where no new capsomers could enter or leave, so increasing the number in this model formulation would improve the analysis of the system. It would give the opportunity to model whether other stable structures might form rather than extra capsids.

An area to investigate would be the effects of crowding, by changing reaction parameters to simulate more packed areas of a cell and more sparse areas, such as those described by Smith *et al* [49]. In their work, most of the parameters necessary to run simulations was provided. The methods described in this thesis allow simulations which model a larger number of capsomers. This means that the assembly of more capsids can be modelled, giving a more representative model of a cell.

One issue is that when increasing the size of the capsid, the number of PSs on the RNA also needed to increase. This led to a large amount of complexity in the number of paths that the RNA could form on the interior of the capsid and a large number of potential kinetic traps that are harder to avoid. This lead simulations to give less efficient assembly that is found in nature, suggesting there is an issue with the model.

In the text, only a very limited number of RNA graphs (in both STNV and the $T = 3$ viruses) had PS distributions optimised, so only a very limited space in a very large parameter space was observed. Due to this, another improvement would be to optimise the RNA for a larger range of $T = 3$ capsids' RNA graphs, aiming to generate the highest possible yield, rather than demonstrating the role of geometry in assembly. There were many RNA graphs that in the initial tests performed more strongly than the ones which had their PS distribution optimised. These RNA graphs could likely have had their PS distributions optimised to give greater yields than those found during this thesis.

In the RNA graphs, each edge is considered to either be in the graph (and so available for the RNA) or not in the graph (and so not available), when the RNA will not necessarily follow such strict rules between each of its PSs. One improvement to the code would be to replace the fixed RNA graph for a capsid, then let each edge on the graph have an associated length and each PS-PS section of the RNA also have an associated length. If the section of the RNA is long enough to reach between these two nodes, then it would be allowed to form a path there. The lengths may need to be altered, if a specific edge is particularly unfavourable (e.g. if internal features of the capsid would block the RNA).

Another potential improvement is related to how strictly the RNA graph is used. Some viruses do not have a neat fraction or multiple of 60 PSs bound to the capsomers. For example, some experiments on MS2 have only identified 54 binding sites [47], suggesting that not all of the 90 capsomers must be bound to PSs, in fact not even all of the AB dimers could be bound. Other viruses will likely follow this pattern, which would allow some unbound PSs to remain unbound, which would increase the range of positions the next bound PS could sit, which would reduce the effect of kinetic traps during assembly. This is not something that the methods described here are capable of modelling, but could potentially have an impact on the results.

During the simulations in this thesis, it is always assumed that two capsomers will bond to each other correctly, in the right orientation, which is not always the case in viral assembly. Altering the code so that defects can form would improve this model. These would likely need a simple, extra state to be added, that occurs with a given probability, where the capsomer can join the intermediate structure with the wrong alignment. This reaction would be reversible but would also impact other capsomers that try to join the structure near to this defect capsomer. As discussed

in Chapter 7, models that can consider defects give results that align better with experimental results [57].

The next suggested improvement involves the RNA. In viral assembly, the RNA is usually produced alongside the CPs, so a system where 2000 fully formed RNAs are present without any CPs is somewhat unrealistic [45]. Whilst the RNA is being formed, it will fold, producing its PSs, so some PSs will be available for capsomers to bind to earlier than others. Allowing the system to model the formation and folding of the RNAs during the assembly simulation would be another way to improve these simulations.

As discussed during Chapter 8, defining all capsids with less than the maximum number of capsomers present in its structure as not assembled is quite a limited metric. This means that two intermediates, for example one with $\frac{2}{60}$ capsomers present and another with $\frac{59}{60}$, are treated as being exactly the same, which is clearly not true. A better way to investigate disassembly would be to include a new definition of when a capsid is disassembled, *e.g.* after a certain percentage of the capsid's capsomers have been removed and then test the effects of altering this percentage. This could result in data similar to other work in this area [5]. Use the same percentage to determine when an assembling capsid is completed too, to see if this affects the hysteresis observed. This would give a more representative idea of the relative stabilities of the structures formed by assembly and disassembly.

In all of the Disassembly simulations, it was assumed that each capsomer-capsomer interaction had the same interaction strength. The RNA was not included in these simulations either. Both of these were shown to have a large effect on the assembly process, so it is likely that they will have large effects on disassembly too. Observing either the effects of non-uniform capsomer-capsomer interaction strengths or the effect of including RNA upon disassembly simulations, would be interesting areas of study. For these, the above criterion for when a capsid can be considered disassembled would be useful. Including RNA in these simulations would need to rely upon using paths for the RNA that were commonly sampled within assembly simulations, such as the one proposed by Wroblewski *et al* [58].

9.2 CONCLUSION

Overall, this thesis describes many ways to model virus assembly and the lim-

itations of these models, in addition to potential improvements that could be implemented. Virus assembly (and assembly of VLPs) is an important and hugely complex topic which will undoubtedly be the focus of research for years to come.

Earlier sections demonstrated the importance of viruses being able to occupy the parameter space close to the tipping point between assembly and disassembly. The inclusion of RNA allows these tipping points to have stability even when the capsomer-capsomer bonds are relatively weak, due to the RNAs stabilising effects. These sections covered the equilibrium states which can take a very long time to be reached, so further work was needed in stochastic simulations, which focused upon the dynamics of approaching this equilibrium state.

Later sections covered larger viruses and investigated how different aspects of the assembly change the yield. The most general result showed that to achieve the highest yields observed, the capsomer-PS binding affinity distribution must have a strong nucleation site, preferably at one end of the RNA. The strength of the interactions between two capsomers also had an impact: too strong and kinetic traps were encountered, too weak and the capsid assembly couldn't get past the nucleation stage. Whether these interactions were uniform or specific to the boundary also impacted the assembly, with some asymmetry across all tilings. The shapes of the capsomers also made a huge impact. Even if they had isomorphic RNA graphs, different tilings always gave different yields. The choice of which edges were present in an RNA graph is more important than how many are present. Some RNA graphs worked better with some tilings than others.

Lastly, when considering disassembly compared to assembly, hysteresis was observed. This demonstrated that reaching equilibrium takes a long time but that viruses will approach it.

— A —

Kite Side Length Proof

When working with T=3 Kite viruses, as mentioned in Section 7.1.2, it is important to know the ratio of side lengths for the kites. In Figure A.1, the two sides have been labelled as α and β, and we want to know the ratio $\frac{\alpha}{\beta}$.

To find this ratio, we start with an equilateral triangle and then we draw a line from the midpoint of each of the triangle's lines to the centre of the triangle. The three initial angles were all 60°, the angles at the bisected points are all 90° and the

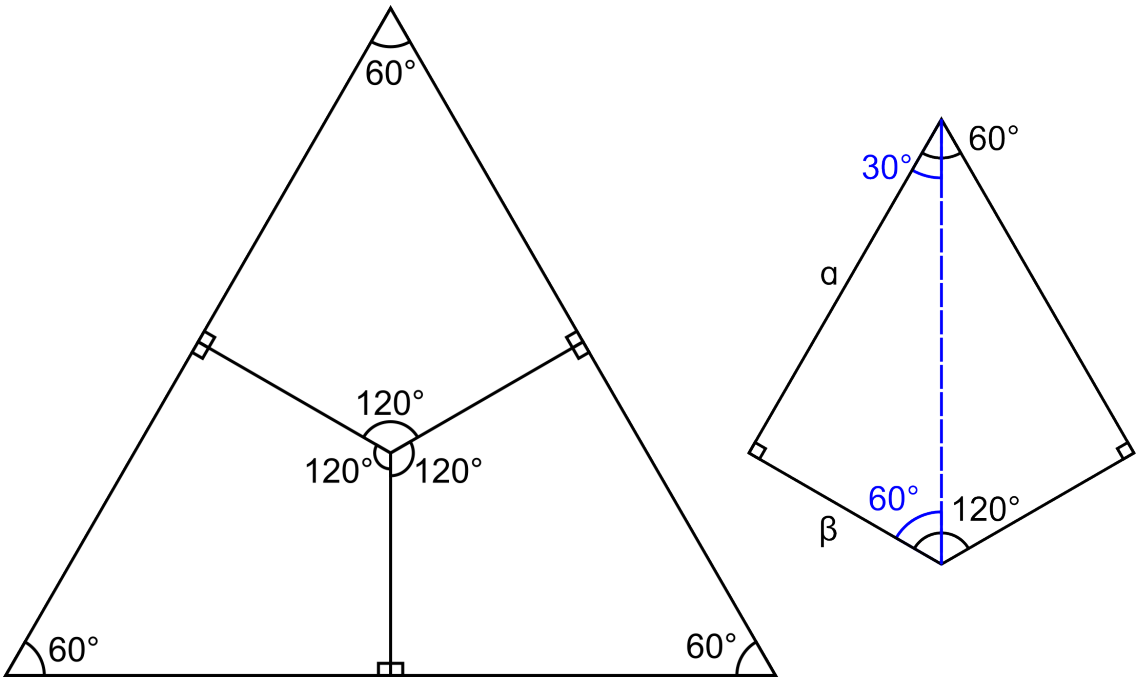

Figure A.1: In T=3 Kite viruses, each triangle on the icosahedron is split into three kite shaped trimers. One such triangle is shown on the left, on the right is one kite, which is bisected (line drawn in blue) along its length.

three central angles are all 120°. Focusing on just one Kite (on the right of Figure A.1), we can bisect the kite along it's length to give a right angled triangle. Using trigonometry, we can then conclude that:

$$\frac{\alpha}{\beta} = \tan(60°) \approx 1.73 \tag{A.1}$$

to give us the ratio we require.

— B —

Algorithms

This appendix is here to compile the various algorithms that are used within Chapter 4 and 5. They are as follows:

Algorithm 3 An algorithm that fills in *mapnn* with all adjacent positions. This is done by taking each connection that was input and adding every permutation of it to the map. An analogous method could be done to fill in *maphp*, *mapexc* and *maptt*, the maps that contain the RNA connections, the mutual exclusion and the tile types for each capsomer position, so they will not be covered.

Variables:
$npro$, the number of capsomers in a complete capsid
$nMaxNeigh$, the maximum number of neighbours for each position
$inNeighbours[5, :]$, stores the inputs, as a list of (FD1, PN1, FD2, PN2, dG_bond)
$connectMap[npro, npro]$, a map of which connections have been added
$mapnn[nMaxNeigh, npro]$, a map of which capsomer positions are connected to which other capsomer positions
$gb[nMaxNeigh, npro]$, stores the ΔG_{bond} for each capsomer-capsomer interaction
$neighbourPointer[npro]$, stores the next position to add a neighbour to *mapnn* for each position
$permutations[60, 122]$, stores the permutations that leave the capsid invariant

procedure ADDNEIGHBOURS
    $connectMap[:] \leftarrow 0$
    $mapnn[:,:] \leftarrow 0$
    $neighbourPointer[:] \leftarrow 1$
    for *list* in *inNeighbours* **do** ▷ Loops over all read in neighbours
        for i in 1, 60 **do** ▷ Loop over all of the permutations
            $newBase \leftarrow$ getNum(*permutations*[i,*list*[1]],*list*[2])
            $newConnect \leftarrow$ getNum(*permutations*[i,*list*[3]],*list*[4])
            if $connectMap[newBase,newConnect] == 0$ **then**
                $mapnn[neighbourPointer[newBase],newBase] \leftarrow newConnect$
                $gb[neighbourPointer[newBase],newBase] \leftarrow list[5]$
                $neighbourPointer[newBase] \leftarrow neighbourPointer[newBase] + 1$
                $connectMap[newBase,newConnect] \leftarrow 1$ ▷ Prevents duplicates
            end if
        end for
    end for
end procedure

Algorithm 4 Algorithm to generate a list of every capsomer's fundamental domain. An analogous method will generate a list of protein capsomer numbers $protNum[npro]$, by swapping what i and j loop over and then swapping each getNum(i,j) with getNum(j,i).

Variables:
$npro$, the number of capsomers needed for one capsid
$protNoMax$, stores the values of Equation 4.1
$FDs[npro]$, a list containing the FD for each position

Functions:
getNum(FD,PN), returns a unique number for each position in the capsid, as detailed in Algorithm 1

```
procedure setFDs
    for i = 1, 60 do
        for j=1, protNoMax(1) do
            FD[getNum(i,j)] ← i
        end for
    end for
    for i = 61, 90 do
        for j=1, protNoMax(2) do
            FD[getNum(i,j)] ← i
        end for
    end for
    for i = 91, 110 do
        for j=1, protNoMax(3) do
            FD[getNum(i,j)] ← i
        end for
    end for
    for i = 111, 122 do
        for j=1, protNoMax(4) do
            FD[getNum(i,j)] ← i
        end for
    end for
end procedure
```

Algorithm 5 A summary of how all of the other algorithms, equations and functions come together to form the module used to initialise all of the graphs that are used during the stochastic simulations of the assembly process.

Maps of the Capsid:
mapnn, a map of which positions are adjacent to each position
maphp, a map of the paths that the RNA can trace a path along
mapexc, a map of mutually exclusive positions for each position
maptt, a map of the tile types for each position

Other Variables:
$protNoMax$, stores the values of Equation 4.1
permutations, stores the permutations
npro, the number of capsomers needed for one capsid
nps, the number of PSs on one RNA
npro_dist, the number of distinct capsomers present
$mutExcNum$, the number of mutually exclusive pairs to read in

procedure makeMap
    Read *permutations* from file
    Read in information from tile file
    Allocate arrays based on information from tile file
    Set up $protNoMax$ from tile file info (as defined in Equation 4.1)
    $npro \leftarrow 60 * protNoMax(1) + 30 * protNoMax(2) + 20 * protNoMax(3) + 12 * protNoMax(4)$
    Using Algorithm 3, set up *mapnn*
    Using a similar method to Algorithm 3, set up *maptt*
    if RNA is used **then**
        Using a similar method to Algorithm 3, set up *maphp*
        if Mutual Exclusion is present **then**
            Using a similar method to Algorithm 3, set up *mapexc*
        end if
    end if
end procedure

Algorithm 6 The program that calculates the number of paths on the RNA graph that visit *nps* vertices and follows all mutual exclusion rules.

Variables:
npro, the number of capsomers in a complete capsid
nps, the number of PSs on one RNA
$mapexc[npro]$, the map of mutual exclusive pairs
$mask[nps, npro]$, a mask that ensures rules on the RNA are applied
$path[nps, nps]$, array that stores the j^{th} position in a path of length i in $path[i, j]$
pathLen, a global store of the length of paths that are desired
counter, a global store of the number of paths of length *pathLen* that exist

Subroutines
nextStep$(mask, path, i)$, as described in Algorithm 7

procedure COUNTPATHS
    Input *pathLen* ▷ Read in length of paths to be counted
    $path[:, :] \leftarrow 0$
    $path[1, 1] \leftarrow 1$ ▷ Paths start at position 1
    $mask[:, :] \leftarrow False$
    $mask[1, 1] \leftarrow True$ ▷ Paths cannot revisit the first position
    if $mapexc[1] \neq 0$ **then**
        $mask[1, mapexc[1]] \leftarrow True$
    end if ▷ When mutual exclusion is used, exclude them from paths
    $counter \leftarrow 0$
    Call nextStep$(mask, path, 1)$ ▷ Start recursive subroutine that sets *counter*
    Output *counter*
end procedure

Algorithm 7 Describes the recursive subroutine nextStep(mask,path,i) that counts all possible pseudo/Hamiltonian paths on a given RNA graph

Variables:
npro, the number of capsomers in a complete capsid
nps, the number of PSs on one RNA
nnps, a list of how many edges each node on the RNA graph has has
maphp, stores the edges of the RNA graph for each position
$mapexc[npro]$, the map of mutual exclusive pairs
$mask[nps, npro]$, a mask that ensures rules on the RNA are applied
$path[nps, nps]$, array that stores the j^{th} position in a path of length i in $path[i, j]$
pathLen, a global store of the length of paths that are desired
counter, a global store of the number of paths of length *pathLen* that exist

procedure NEXTSTEP($mask, path, i$)
  if $i = pathLen$ **then** ▷ Test if the path is long enough and tally it if so
    $counter \leftarrow counter + 1$
  else
    for $k = 1, nnps[path[i, i]]$ **do** ▷ Loop over all current neighbours
      $mask[i + 1, :] = False$
      $path[i + 1, :] = 0$
      $nextPoint \leftarrow maphp[k, path[i, i]]$
      if NOT $mask[i, nextPoint]$ **then** ▷ Use $mask$ to stop invalid paths
        for $l = 1, i$ **do** ▷ Update next step's variables
          $path[i + 1, l] \leftarrow path[i, l]$
        end for
        $mask[i + 1, :] \leftarrow mask[i, :]$
        $mask[i + 1, nextPoint] \leftarrow True$
        if $mapexc[nextPoint] \neq 0$ **then**
          $mask[i + 1, mapexc[nextPoint]] \leftarrow True$
        end if
        $path[i + 1, i + 1] \leftarrow nextPoint$
        CALL nextStep$[mask, path, i + 1]$ ▷ Recurse until paths are done
      end if
    end for
  end if
end procedure

Algorithm 8 A method to read RNA paths from a file and to permute them to allow easier analysis of the paths.

Variables:
$filelength$, the number of paths to be input in the algorithm
$npro$, the number of capsomers in a complete capsid
nps, the number of PSs on one RNA
$permutations[60, 122]$, stores the permutations that leave the capsid invariant
$pathPerms[nps, 60 * filelength]$, list of the permutated paths to be used later
$FDs[npro]$, list of the FDs that each capsomer sits in
$protNum[npro]$, list of the protein number for each capsomer

Functions:
getNum(FD,PN), returns a unique number for each position in the capsid, as detailed in Algorithm 1

```
procedure PERMUTEPATHS
    for i = 1, filelength do                                  ▷ loop over all lines
        Read line from file into inPath[1 : nps]
        for j = 1, 60 do                                ▷ Loops over all permutations
            for k = 1, nps do                                    ▷ Loops over all PSs
                if inPath[k] ≤ 0 then
                    pathPerms[k, 60(i − 1) + j] ← 0
                    CYCLE        ▷ Allows analysis of incomplete capsids if desired
                end if
                p ← inPath[k]                              ▷ p is current position in path
                permFD ← permutations[j, FDs[p]]                    ▷ permute p's FD
                pathPerms[k, 60(i − 1) + j] ← getNum(permFD, protNum[p])
            end for
        end for
    end for
end procedure
```

Algorithm 9 A method that takes the output of Algorithm 8 and produces a tally of which RNA paths are most common.

Variables:
nps, the number of PSs on one RNA
$filelength$, the number of paths that were input
c, the minimum number of times a path must occur for it to be recorded
$pathPerms[nps, 60 * filelength]$, a list of the input paths and their permutations
$nRepeats$, the number of paths that are repeated at least c times
$repPaths[nps, FLOOR(filelength/c)]$, a list of the paths with at least c repeats
$pathCount[FLOOR(filelength/c)]$, a list of the number of times that the corresponding path in $repPaths$ has been repeated
$marker[60 * filelength]$, a mask so that paths are only counted once

procedure PATHTALLY
  $marker[:] \leftarrow 0$ ▷ Set variables to initial values
  $repPaths[:,:] \leftarrow 0$
  $pathCount[:] \leftarrow 0$
  $nRepeats \leftarrow 1$
  for $i = 1, 60(filelength - 1), 60$ **do** ▷ Consider first permutation of each path
    if $marker[i] == 1$ **then** CYCLE ▷ Skip if it's already been counted
    tally = 1
    for $j = i + 60, 60 * filelength$ **do** ▷ Check permutations of all later paths
      if $marker[60 * FLOOR(\frac{j-1}{60}) + 1] == 1$ **then** CYCLE
      match = 1
      for $k = 1, nps$ **do**
        if $pathPerms[k, i] \neq pathPerms[k, j]$ **then**
          $match \leftarrow 0$
          EXIT loop
        end if ▷ Skip rest of path if anything doesn't match
      end for
      if $match == 1$ **then** ▷ Increment tally if the paths match
        $tally \leftarrow tally + 1$
        $marker[60 * FLOOR(\frac{j-1}{60}) + 1] \leftarrow 1$ ▷ Mark counted path
      end if
    end for
    if $tally \geq c$ **then** ▷ Only record path if it's repeated enough
      $repPaths[:, nRepeats] \leftarrow pathPerms[:, i]$
      $pathCount[nRepeats] \leftarrow tally$
      $nRepeats \leftarrow nRepeats + 1$
    end if
    Perform Insertion Sort on lists using the values of $pathCount$
  end for
end procedure ▷ Results in sorted list of most common paths

Algorithm 10 Function that is given coordinates and a rotation instruction, which outputs the rotated coordinates

Variables:
rot, the instruction for what rotation should be done
$iCoord[3]$, input coordinates
$gr = \frac{1+\sqrt{5}}{2}$, the golden ratio
$k = \frac{1}{\sqrt{3}}$, a normalisation factor
$ax\_5[:] = \frac{1}{\sqrt{2+gr}}(0, 1, gr)$, the 5-fold axis' coordinates
$ax\_3[:] = k(1, 1, 1)$, the 3-fold axis' coordinates
$theta$, the angle that the coordinates are rotated by
$axis[3]$, the axis that the coordinates are rotated about
$RM[3, 3]$ an array, as defined by the Matrix in Equation 5.1
$oCoord[3]$, output coordinates

procedure MPRODUCT($rot, iCoord$)
  if $rot = 1$ **then** ▷ Clockwise about 5-fold axis
    $theta \leftarrow \frac{-2\pi}{5}$
    $axis[:] \leftarrow ax\_5[:]$
  else if $rot = 2$ **then** ▷ Anti-clockwise about 5-fold axis
    $theta \leftarrow \frac{2\pi}{5}$
    $axis[:] \leftarrow ax\_5[:]$
  else if $rot = 3$ **then** ▷ Clockwise about 3-fold axis
    $theta \leftarrow \frac{-2\pi}{3}$
    $axis[:] \leftarrow ax\_3[:]$
  else if $rot = 4$ **then** ▷ Anti-clockwise about 3-fold axis
    $theta \leftarrow \frac{2\pi}{3}$
    $axis[:] \leftarrow ax\_3[:]$
  end if
  Setup RM from Equation 5.1 using $axis$ and $theta$
  $oCoord[1] \leftarrow RM[1,1]*iCoord[1]+RM[1,2]*iCoord[2]+RM[1,3]*iCoord[3]$
  $oCoord[2] \leftarrow RM[2,1]*iCoord[1]+RM[2,2]*iCoord[2]+RM[2,3]*iCoord[3]$
  $oCoord[3] \leftarrow RM[3,1]*iCoord[1]+RM[3,2]*iCoord[2]+RM[3,3]*iCoord[3]$
  Return $oCoord$ ▷ Gives result of rotation back
end procedure

Algorithm 11 Reads in the coordinates and the rotations that are used to generate the coordinates for each capsomer position.

Variables:
$npro$, the number of capsomers in a complete capsid
$FDs[npro]$, a list containing the FD for each position
$protNo[npro]$, a list containing the protein number for each position
$coords[3, npro]$, the coordinates for each position
$lineLength$, the maximum number of rotational instructions
$instr[lineLength, 122]$, stores the rotation instructions for each FD/axis

Functions:
getNum(FD, PN), defined in Algorithm 1
MProduct$(rot, iCoord)$, defined in Algorithm 10

procedure
    Read in the FD, PN and coordinates for each distinct protein from file
    Store all of those coordinates in $coords[getNum(FD, PN), :]$
    Read in instructions from file to $instr[:, :]$
    for $i = 1, npro$ **do** ▷ Finds coords for all of the positions
        if $FDs[i] < 61$ **then**
            $originalNo \leftarrow getNum(1, protNo[i])$
        else if $FDs[i] < 91$ **then**
            $originalNo \leftarrow getNum(61, protNo[i])$
        else if $FDs[i] < 111$ **then**
            $originalNo \leftarrow getNum(91, protNo[i])$
        else
            $originalNo \leftarrow getNum(111, protNo[i])$
        end if
        $coords[:, i] \leftarrow coords[:, originalNo]$ ▷ Sets coords to the non-rotated value
        for $j = 1, lineLength$ **do** ▷ Completes all rotations so coords is correct
            if $instr[j, FDs[i]] \neq 0$ **then**
                $coords[:, i] \leftarrow MProduct(instr[j, FDs[i]], coords[:, i])$
            else
                Exit loop
            end if
        end for
    end for
end procedure

Algorithm 12 This algorithm describes how the code loops through the graphs that represent the capsid and outputs them in a 3D representation. It creates a .bild file and within these files, the main useful elements of these files are either cylinders or spheres, which can be used to represent edges and vertices respectively.

Variables:
$npro$, the number of capsomers in a complete capsid
$coords[3, npro]$, the coordinates for each position
$nnps[npro]$, stores the number of edges for each vertex in the RNA graph
$maphp$, stores the edges for each vertex in the RNA graph
$nncp[npro]$, stores the number of edges for each vertex in the capsomer graph
$mapnn$, stores the edges for each vertex in the capsomer graph

procedure
  for $i = 1, npro$ **do** ▷ Loops over all positions
    Output a sphere at $coords[:, i]$
  end for
  for $i = 1, npro$ **do**
    for $j = 1, nnps[i]$ **do** ▷ Loops over all RNA edges for all positions
      if $i < maphp[j, i]$ **then** ▷ Prevents duplicates
        Output cylinder from $coords[:, i]$ to $coords[:, maphp[j, i]]$
      end if
    end for
    for $j = 1, nncp[i]$ **do** ▷ Loops over all capsomer interactions for all positions
      if $i < mapnn[j, i]$ **then** ▷ Prevents duplicates
        Output cylinder from $coords[:, i]$ to $coords[:, mapnn[j, i]]$
      end if
    end for
  end for
end procedure

Algorithm 13 This algorithm describes how the code loops through a path and outputs a 3D image of the path the RNA traces out. It creates a .bild file and within these files, the main useful elements of these files are either cylinders or spheres, which can be used to represent edges and vertices respectively.

Variables:
nps, the number of PSs on one RNA
$coords[3, npro]$, the coordinates for each position
$path[nps]$, stores the path as a series of integer positions

procedure
    Read in $path[:]$ from file
    for $j = 1, nps - 1$ **do** ▷ Loops over all pairs of consecutive positions
        if $path[j] \leq 0$ **then** CYCLE ▷ Skips any missing positions
        Output sphere at $coords[:, path[j]]$
        if $path[j + 1] \leq 0$ **then** CYCLE ▷ Stops before end
        Output cylinder from $coords[:, path[j]]$ to $coords[:, path[j + 1]]$
    end for
    if $path[nps] > 0$ **then** Output sphere at $coords[:, path[j + 1]]$
end procedure

— C —

Tile File Definitions

In Section 4.2, the text mentions a file that is used for inputting the information on the capsid that is needed to simulate it. In this appendix, this file will be defined and a few examples of how they can be generated is provided.

These files are intended to both have the necessary information for the algorithm but also be easily readable and generated by a person. This means that a small number of lines are reserved for labels that will show separation between sections and make it easier for people to read. Earlier lines will define the number of lines that the algorithm needs to read in too. The format used for these tile file is defined as follows:

1 #DP, #PS, #ME

The number of distinct protein capsomers in the capsid, the number of packaging signals along the length of the RNA and the number of mutually exclusive position pairs to be read in for the RNA

2 #NB[1:#DP]

A list containing the number of Neighbour Interactions for each of the distinct protein capsomers

3 #RC[1:#DP]

A list containing the number of RNA Connections for each distinct protein capsomer. Set as 0 if the RNA is not able to bind to a capsomer.

4 "TILE INFO AND BONDS"

Label for reading

5 "FD1 / PN1 / FD2 / PN2 / DELTA G"

Column headings for reading

6-... FD1, PN1, FD2, PN2, dG

A series of $\sum_{i=1}^{\#DP}(\#NB[i])$ *lines which each provide a neighbour interaction for the capsid, as the fundamental domain and protein number for each capsomer involved in the interaction, followed by the strength of the interaction they form together*

1' "TILE TYPES FOR ASSEMBLY"

Label for reading

2' "FD / PN / TYPE"

Column headings for reading

3'-... FD, PN, PT

A series of #DP lines, each of which define the tile type of a given protein capsomer, defined via its fundamental domain and protein number, followed by which type of tile the capsomer is

1" "RNA PATH MOVES"

Label for reading

2" "FD / PN / FD / PN"

Column headings for reading

3"-... FD, PN, FD, PN

A series of $\sum_{i=1}^{\#PS}(\#RC[i])$ *lines which each provide an RNA connections, i.e. the capsomer's where the adjacent PSs on the RNA can sit if one PS is bound to the initial capsomer. Listed as a fundamental domain and protein number for the first capsomer and then again for the second.*

1"' "Mutual Exclusive Pairs"

Label for reading

2"' "FD / PN / FD / PN"

Column headings for reading

3"'-... FD, PN, FD, PN

A series of #ME lines which each provide a mutually exclusive pair of capsomers, so if a PS is bound to one capsomer of a pair, its partner cannot be bound to a PS. Listed as a fundamental domain and protein number for the first capsomer and then again for the second.

Those were a list of all the lines in the tile file and states what they should contain, alongside the reason why they are used. Next, the way to produce these files is covered.

C.1 HOW TO CREATE TILE FILES

To demonstrate how tile files are produced and what the resultant graphs from running this algorithm look like, a few examples are provided below. They are laid out on an icosahedral net in Figure C.1 (*cf* from Figure 4.3), the key lines of the tile files follow in Appendix C.2. The numbering from Figure C.2 (*cf* from Figure 4.6) is used to label the positions that the capsomers sit in.

Consider Figure C.1's Example 1. There are three distinct capsomers, shown in red, yellow and green. None of them bind to RNA and so cannot have any mutual exclusion.We arbitrarily choose that green represents the first capsomer that sits in each FD and yellow the second. Red represents the pentagonal capsomer that sits on the 5-fold axis. Green and yellow capsomers have 4 interactions each and red ones have 10 interactions each.Cross referencing Figures C.1 and C.2 gives the fundamental domains and protein capsomer numbers for each interaction, which is paired with the interaction strength dG of that interaction. Each interaction needs to be written from the perspective of each capsomer. This means some care is needed to ensure both are present in the tile file but this makes it easier to count how many interactions are present for each distinct capsomer. When the algorithm reads that there is a interaction between A and B, it only adds this edge to the list of edges for vertex A, so this will not result in double counting (also, the algorithm checks the current edges before adding new ones, so it avoids duplicates there too). The algorithm can handle any FDs being used for each interaction (it doesn't always need to be in FD 1) though it is sensible to do each interaction for a distinct capsomer from the perspective of one specific capsomer to prevent accidental duplicates/missing interactions. The ΔG_{bond} value does not need to be the same for every interaction, though it should be symmetric (*i.e.* if capsomer 1 binds to capsomer 2 with strength

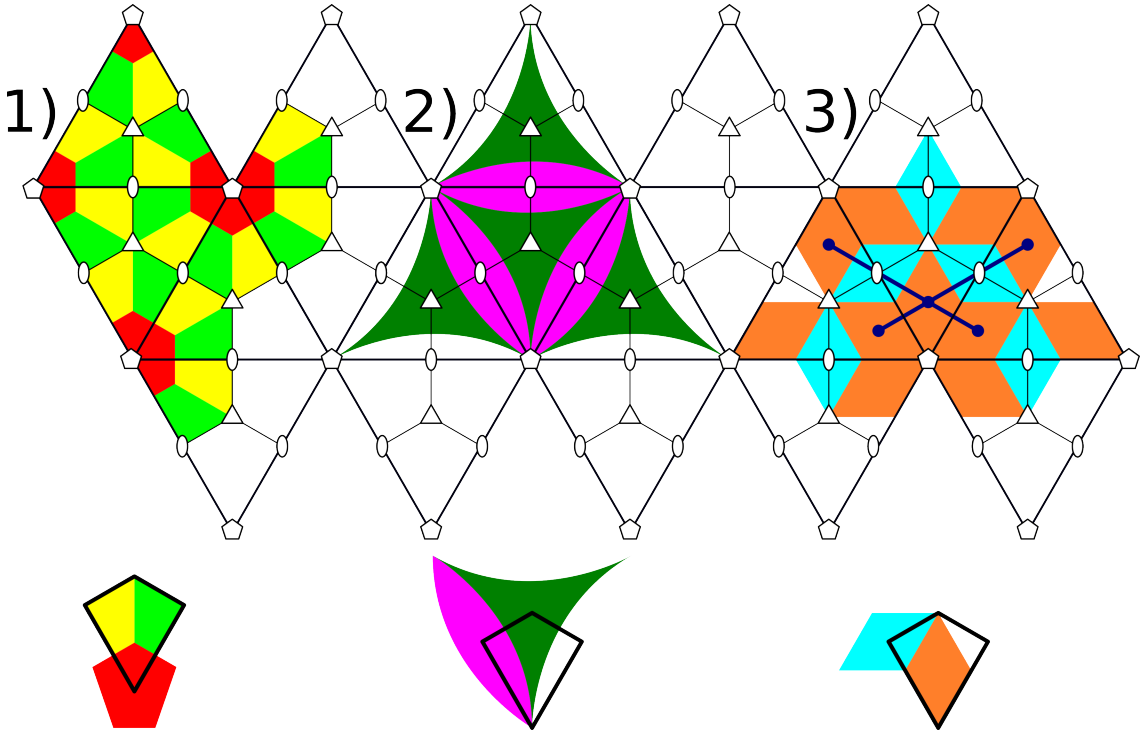

Figure C.1: *cf.* from Figure 4.3. Three examples of potential tilings of the capsid, shown as patches of tilings, to show how little information is needed for tile files. Below each patch is an asymmetric unit, with one copy of each distinct capsomer shown, including ones that sit upon symmetry axes. The algorithm would only need the information for these capsomers and could then generalise it to the entire capsid, using icosahedral symmetry. Example 1 is an example where there are multiple capsomers present in one FD. Example 2 shows an example where all capsomers sit on symmetry axes (specifically the 2-fold and 3-fold axes). Example 3 illustrates how an average rhombic $T = 3$ virus would be tiled, including RNA that can go around the 5-fold axis and across both nearby 2-fold axes. The navy lines indicate the four edges on one vertex of a potential RNA graph. These lines could be extended to cover the whole capsid and thus form the RNA graph for the capsid. In Appendix C, it is detailed how these become tile files for the algorithm to understand. We shall assume that Examples 1 and 2 do not include RNA.

dG, then capsomer 2 should bind to capsomer 1 with strength dG). The effects on assembly of changing interaction strengths independently was covered in Sections 7.1.2 and 7.2.6. For Example 1, let the three distinct capsomers be made of different tile types (this should normally be decided upon by using experimental data but as this is not a real virus, none exists so we follow this assumption, which is reasonable as they are all different shapes), which is also input via the tile file. The numbers used for tile types must all be integers beginning at 1 and ending at the number of different tile types present in the capsid however the order is arbitrary.

Next, consider Example 2, which is also an RNA-free assembly. It features two capsomers, green and magenta, that each form a different number of interactions. In this case, every interaction is rotationally equivalent to every other interaction, so the values of ΔG_{bond} must be the same for this example. This example shows that there is no need for there to be a capsomer in a FD for this to work. The two distinct capsomers are different shapes so they get different tile types. The Tile File for this is given in Appendix C.

Finally, consider Example 3. Here there are two distinct capsomers, orange and cyan. The RNA is able to visit 60 capsomers and there is no mutual exclusion (covered in more detail in Section 6.2). Both capsomers will form four interactions but only one is able to bind to RNA. Then, the interactions are generated, as before in part (a). As the capsomers are all rhombic shapes, they may (and for this example, we assume they will) be made of the same tile type, so they are given the same number in lines 3' and 4'.

Finally, there are the edges for the RNA graph, input similarly to the capsomer-capsomer contacts, only without the interaction strengths. Often, the capsomer graph and the RNA graph will have some matching edges but this is not required and usually not all of them do, as in this example. The Tile Files for these three examples are also given in Appendix C.2.

With these tile files, the algorithm generates both a capsomer and an RNA graph (when RNA is present) to represent the capsid. They are stored as 2D arrays which, for each vertex, store a list of the vertices that are adjacent to the current vertex. This means that the neighbours can be found quickly, without needing to search through a sparse matrix.

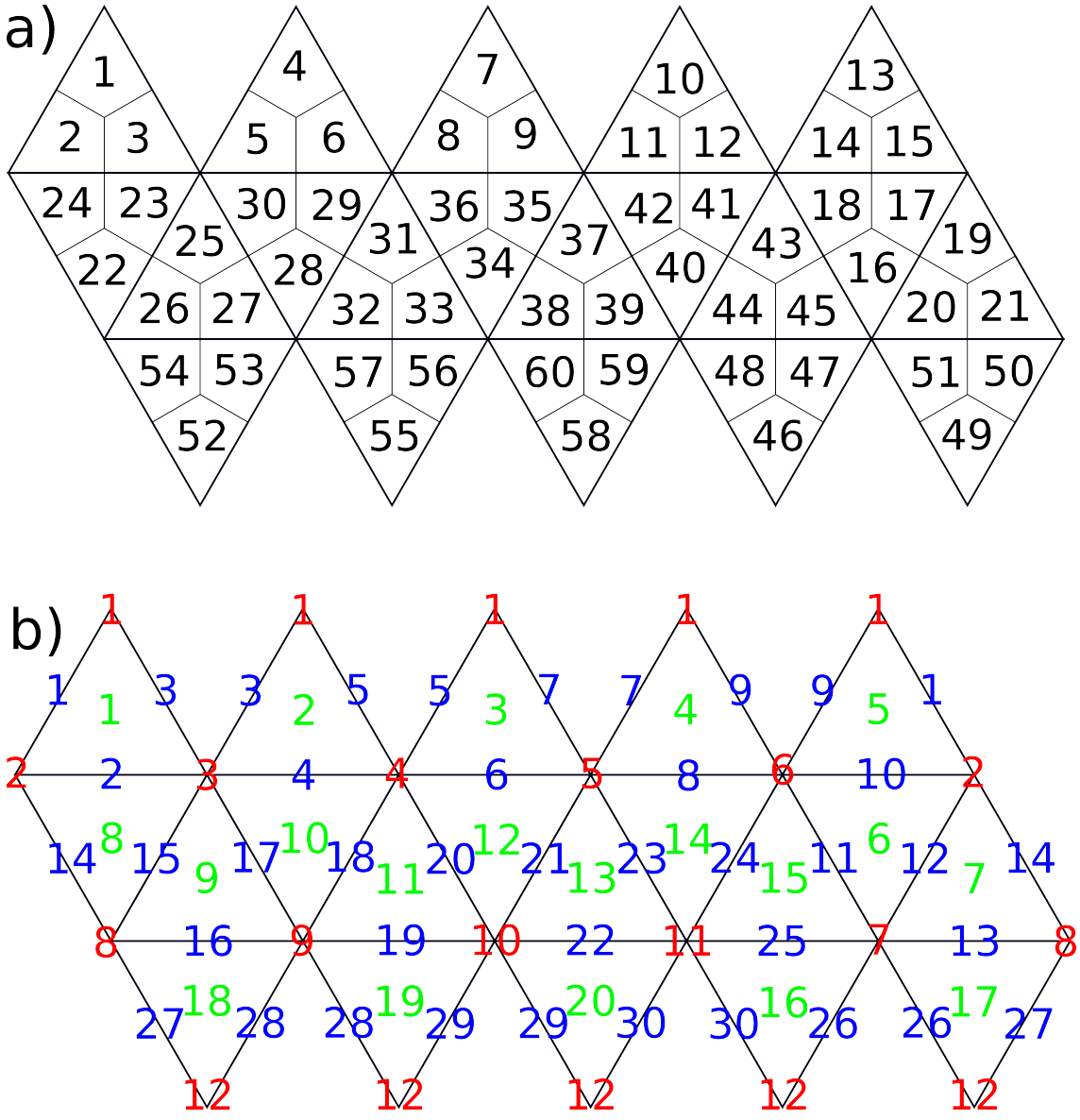

Figure C.2: *cf.* from Figure 4.6. Planar nets of the icosahedron, split up so that each fundamental domain and axis can be numbered. There are 2-fold axes at the midpoint of edges, 3-fold axes at the centre of the triangles and 5-fold axes at each vertex. Fundamental domains have been chosen as kite shapes for the purpose of this numbering. (a) Shows the Fundamental Domains and assigns a number to each of them. (b) Numbers the 5-fold (red), 3-fold (green) and 2-fold (blue) symmetry axes. When multiple positions have the same number in the same colour, this just indicates that when the net is folded, they are in the same position. The two figures line up, so fundamental domain 1 would sit between 5-fold 1, 3-fold 1 and 2-folds 1 and 3.

C.2 TILE FILE EXAMPLES

In Section C.1, how to generate tile files was covered. Below are the key lines on the files for the three examples discussed in that Section. The two relevant figures (4.6, 4.3) have been copied into this Appendix as Figures C.2 and C.1 for convenience. The positions of the capsomers in the second figure correlate directly to the numbering given in the first figure, which should make it easier to follow the method used.

First, for Example 1 the file is:

1) 3, 0, 0
2) 4, 4, 10
3) 0, 0, 0

...

6) 23, 1, 23, 2, ΔG_{bond}
7) 23, 1, 3, 2, ΔG_{bond}
8) 23, 1, 113, 1, ΔG_{bond}
9) 23, 1, 24, 1, ΔG_{bond}
10) 23, 2, 23, 1, ΔG_{bond}
11) 23, 2, 25, 1, ΔG_{bond}
12) 23, 2, 113, 1, ΔG_{bond}
13) 23, 2, 24, 2, ΔG_{bond}
14) 113, 1, 3, 1, ΔG_{bond}
15) 113, 1, 3, 2, ΔG_{bond}
16) 113, 1, 5, 1, ΔG_{bond}
17) 113, 1, 5,2, ΔG_{bond}
18) 113, 1, 30, 1, ΔG_{bond}
19) 113, 1, 30,2, ΔG_{bond}
20) 113, 1, 25, 1, ΔG_{bond}
21) 113, 1, 25, 2, ΔG_{bond}
22) 113, 1, 23, 1, ΔG_{bond}
23) 113, 1, 23, 2, ΔG_{bond}

...

3') 23, 1, 1

4') 23, 2, 2

5') 113, 1, 3

Next, for Example 2 the file is:

1) 2, 0, 0

2) 3, 2

3) 0, 0

...

6) 102, 1, 66, 1, ΔG_{bond}

7) 102, 1, 80, 1, ΔG_{bond}

8) 102, 1, 81, 1, ΔG_{bond}

9) 66, 1, 93, 1, ΔG_{bond}

10) 66, 1, 102, 1, ΔG_{bond}

...

3') 102, 1, 1

4') 66, 1, 2

Last, for Example 3 the file is:

1) 2, 60, 0

2) 4, 4

3) 4, 0

...

6) 16, 1, 20, 1, ΔG_{bond}

7) 16, 1, 45, 1, ΔG_{bond}

8) 16, 1, 71, 1, ΔG_{bond}

9) 16, 1, 72, 1, ΔG_{bond}

10) 71, 1, 16, 1, ΔG_{bond}

11) 71, 1, 18, 1, ΔG_{bond}

12) 71, 1, 43, 1, ΔG_{bond}

13) 71, 1, 45, 1, ΔG_{bond}

...

3') 16, 1, 1

4') 71, 1, 1

...

3") 16, 1, 20, 1

4") 16, 1, 45, 1

5") 16, 1, 19, 1

6") 16, 1, 43, 1

References

[1] D. F. Anderson. "A modified next reaction method for simulating chemical systems with time dependent propensities and delays". In: *The Journal of Chemical Physics* 127.21 (Dec. 2007), p. 214107. ISSN: 0021-9606. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.2799998/15406018/214107_1_online.pdf.

[2] E. Anderson et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8 (paperback).

[3] e. Baron S. *Medical Microbiology, 4th Edition*. Galveston (TX): University of Texas Medical Branch at Galveston, 1996. eprint: https://www.ncbi.nlm.nih.gov/books/NBK7627/.

[4] A. P. Biela, A. Naskalska, F. Fatehi, R. Twarock, and J. G. Heddle. "Programmable polymorphism of a virus-like particle". In: *Communications Materials* 3.1 (Feb. 2022), p. 7. ISSN: 2662-4443.

[5] N. E. Brunk and R. Twarock. "Percolation Theory Reveals Biophysical Properties of Virus-like Particles". In: *ACS Nano* 15.8 (2021). PMID: 34296852, pp. 12988–12995. eprint: https://doi.org/10.1021/acsnano.1c01882.

[6] D. H. Bunka, S. W. Lane, C. L. Lane, E. C. Dykeman, R. J. Ford, A. M. Barker, R. Twarock, S. E. Phillips, and P. G. Stockley. "Degenerate RNA Packaging Signals in the Genome of Satellite Tobacco Necrosis Virus: Implications for the Assembly of a T=1 Capsid". In: *Journal of Molecular Biology* 413.1 (2011), pp. 51–65. ISSN: 0022-2836.

[7] Y. Cao, D. T. Gillespie, and L. R. Petzold. "Efficient step size selection for the tau-leaping simulation method". In: *The Journal of Chemical Physics* 124.4 (Jan. 2006), p. 044109. ISSN: 0021-9606. eprint: https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.2159468/15382239/044109_1_online.pdf.

[8] Y. Cao, H. Li, and L. Petzold. "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems". In: *The Journal of Chemical Physics* 121.9 (Sept. 2004), pp. 4059–4067. ISSN: 0021-9606. eprint: https://pubs.aip.org/aip/jcp/article-pdf/121/9/4059/19310270/4059_1_online.pdf.

[9] D. L. D. Caspar and A. Klug. "Physical Principles in the Construction of Regular Viruses". In: *Cold Spring Harbor Symposia on Quantitative Biology* 27.0 (Jan. 1962), pp. 1–24.

[10] C. D. I. A. CBE. *Foot and Mouth Disease 2007: A Review*. URL: https://assets.publishing.service.gov.uk/media/5a7c7289e5274a5255bceb57/0312.pdf. (accessed: 04/07/2024).

[11] CDC. *The Deadliest Flu: The Complete Story of the Discovery and Reconstruction of the 1918 Pandemic Virus*. URL: https://archive.cdc.gov/#/details?url=https://www.cdc.gov/flu/pandemic-resources/reconstruction-1918-virus.html. (accessed: 03/06/2024).

[12] H. Y. Chen, M. D. Mascio, A. S. Perelson, D. D. Ho, and L. Zhang. "Determination of virus burst size in vivo using a single-cycle SIV in rhesus macaques". In: *Proceedings of the National Academy of Sciences* 104.48 (2007), pp. 19079–19084. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.0707449104.

[13] X. Dai, Z. Li, M. Lai, S. Shu, Y. Du, H. Zhou, and R. Sun. "In situ structures of the genome and genome-delivery apparatus in a single-stranded RNA virus". In: *Nature* 541 (Dec. 2016).

[14] K. Damodaran, V. S. Reddy, J. E. Johnson, and C. L. Brooks. "A General Method to Quantify Quasi-equivalence in Icosahedral Viruses". In: *Journal of Molecular Biology* 324.4 (2002), pp. 723–737. ISSN: 0022-2836.

[15] A. Düx et al. "The history of measles: from a 1912 genome to an antique origin". In: *bioRxiv* (2019). eprint: https://www.biorxiv.org/content/early/2019/12/30/2019.12.29.889667.full.pdf.

[16] E. Dykeman, N. Grayson, K. Toropova, N. Ranson, P. Stockley, and R. Twarock. "Simple Rules for Efficient Assembly Predict the Layout of a Packaged Viral RNA". In: *Journal of Molecular Biology* 408.3 (2011), pp. 399–407. ISSN: 0022-2836.

[17] E. C. Dykeman. "Modelling ribosome kinetics and translational control on dynamic mRNA". In: *PLOS Computational Biology* 19.1 (Jan. 2023), pp. 1–20.

[18] E. C. Dykeman, P. G. Stockley, and R. Twarock. "Building a viral capsid in the presence of genomic RNA". In: *Phys. Rev. E* 87 (2 Feb. 2013), p. 022717.

[19] E. C. Dykeman, P. G. Stockley, and R. Twarock. "Solving a Levinthal's paradox for virus assembly identifies a unique antiviral strategy". In: *Proceedings of the National Academy of Sciences* 111.14 (2014), pp. 5361–5366. eprint: `https://www.pnas.org/doi/pdf/10.1073/pnas.1319479111`.

[20] E. C. Dykeman. "A stochastic model for simulating ribosome kinetics in vivo". In: *PLOS Computational Biology* 16.2 (Feb. 2020), pp. 1–20.

[21] F. Fatehi and R. Twarock. "An interaction network approach predicts protein cage architectures in bionanotechnology". In: *Proceedings of the National Academy of Sciences* 120.50 (2023), e2303580120. eprint: `https://www.pnas.org/doi/pdf/10.1073/pnas.2303580120`.

[22] R. Ford, A. Barker, S. Bakker, R. Coutts, N. Ranson, S. Phillips, A. Pearson, and P. Stockley. "Sequence-Specific, RNA–Protein Interactions Overcome Electrostatic Barriers Preventing Assembly of Satellite Tobacco Necrosis Virus Coat Protein". In: *Journal of molecular biology* 425 (Jan. 2013).

[23] R. F. Garmann, A. M. Goldfain, C. R. Tanimoto, C. E. Beren, F. F. Vasquez, D. A. Villarreal, C. M. Knobler, W. M. Gelbart, and V. N. Manoharan. "Single-particle studies of the effects of RNA–protein interactions on the self-assembly of RNA virus particles". In: *Proceedings of the National Academy of Sciences* 119.39 (2022), e2206292119. eprint: `https://www.pnas.org/doi/pdf/10.1073/pnas.2206292119`.

[24] J. A. Geraets. "Self-assembling nanoscale systems". Sept. 2015.

[25] J. A. Geraets, E. C. Dykeman, P. G. Stockley, N. A. Ranson, and R. Twarock. "Asymmetric Genome Organization in an RNA Virus Revealed via Graph-Theoretical Analysis of Tomographic Data". In: *PLOS Computational Biology* 11.3 (Mar. 2015), pp. 1–16.

[26] M. A. Gibson and J. Bruck. "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels". In: *The Journal of Physical Chemistry A* 104.9 (2000), pp. 1876–1889. eprint: https://doi.org/10.1021/jp993732q.

[27] D. T. Gillespie. "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions". In: *Journal of Computational Physics* 22.4 (1976), pp. 403–434. ISSN: 0021-9991.

[28] D. T. Gillespie. "Exact stochastic simulation of coupled chemical reactions". In: *The Journal of Physical Chemistry* 81.25 (1977), pp. 2340–2361. eprint: https://doi.org/10.1021/j100540a008.

[29] D. T. Gillespie. "Stochastic Simulation of Chemical Kinetics". In: *Annual Review of Physical Chemistry* 58.1 (2007). PMID: 17037977, pp. 35–55. eprint: https://doi.org/10.1146/annurev.physchem.58.032806.104637.

[30] D. T. Gillespie and L. R. Petzold. "Improved leap-size selection for accelerated stochastic simulation". In: *The Journal of Chemical Physics* 119.16 (Oct. 2003), pp. 8229–8234. ISSN: 0021-9606. eprint: https://pubs.aip.org/aip/jcp/article-pdf/119/16/8229/10850811/8229_1_online.pdf.

[31] D. A. Henderson. "The eradication of smallpox – An overview of the past, present, and future". In: *Vaccine* 29 (2011). Smallpox Eradication after 30 Years: Lessons, Legacies and Innovations, pp. D7–D9. ISSN: 0264-410X.

[32] S. R. Hill, R. Twarock, and E. C. Dykeman. "The impact of local assembly rules on RNA packaging in a T = 1 satellite plant virus". In: *PLOS Computational Biology* 17.8 (Aug. 2021), pp. 1–18.

[33] A. Klug and D. Caspar. "The Structure of Small Viruses". In: ed. by K. M. Smith and M. A. Lauffer. Vol. 7. Advances in Virus Research. Academic Press, 1961, pp. 225–325.

[34] S. W. Lane, C. A. Dennis, C. L. Lane, C. H. Trinh, P. J. Rizkallah, P. G. Stockley, and S. E. Phillips. "Construction and Crystal Structure of Recombinant STNV Capsids". In: *Journal of Molecular Biology* 413.1 (2011), pp. 41–50. ISSN: 0022-2836.

[35] S. W. Lane, C. A. Dennis, C. L. Lane, C. H. Trinh, P. J. Rizkallah, P. G. Stockley, and S. E. Phillips. "Construction and Crystal Structure of Recombinant STNV Capsids". In: *Journal of Molecular Biology* 413.1 (2011), pp. 41–50. ISSN: 0022-2836.

[36] H. Li and L. Petzold. "Logarithmic direct method for discrete stochastic simulation of chemically reacting systems". In: *Journal of Chemical Physics* (Aug. 2006).

[37] J. Louten. "Chapter 2 - Virus Structure and Classification". In: *Essential Human Virology*. Ed. by J. Louten. Boston: Academic Press, 2016, pp. 19–29. ISBN: 978-0-12-800947-5.

[38] E. Mathieu et al. "Coronavirus Pandemic (COVID-19)". In: *Our World in Data* (2020). https://ourworldindata.org/coronavirus.

[39] J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, and N. F. Samatova. "The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior". In: *Computational Biology and Chemistry* 30.1 (2006), pp. 39–49. ISSN: 1476-9271.

[40] W. H. McNeill. *Plagues and Peoples*. URL: `https://web.archive.org/web/20091003164005/http://www.birdflubook.com/a.php?id=40`. (accessed: 09/10/2009).

[41] D. Montiel-Garcia, N. Santoyo-Rivera, P. Ho, M. Carrillo-Tripp, C. L. B. III, J. E. Johnson, and V. S. Reddy. "VIPERdb v3.0: a structure-based data analytics platform for viral capsids". In: *Nucleic Acids Research* 49.D1 (Dec. 2020), pp. D809–D816. ISSN: 0305-1048. eprint: `https://academic.oup.com/nar/article-pdf/49/D1/D809/35364004/gkaa1096.pdf`.

[42] N. Patel, E. C. Dykeman, R. H. A. Coutts, G. P. Lomonossoff, D. J. Rowlands, S. E. V. Phillips, N. Ranson, R. Twarock, R. Tuma, and P. G. Stockley. "Revealing the density of encoded functions in a viral RNA". In: *Proceedings of the National Academy of Sciences* 112.7 (2015), pp. 2227–2232. eprint: `https://www.pnas.org/doi/pdf/10.1073/pnas.1420812112`.

[43] N. Patel, E. Wroblewski, G. Leonov, S. E. V. Phillips, R. Tuma, R. Twarock, and P. G. Stockley. "Rewriting nature's assembly manual for a ssRNA virus". In: *Proceedings of the National Academy of Sciences* 114.46 (2017), pp. 12255–12260. eprint: `https://www.pnas.org/doi/pdf/10.1073/pnas.1706951114`.

[44] C. Pérez-Segura, B. C. Goh, and J. A. Hadden-Perilla. "All-Atom MD Simulations of the HBV Capsid Complexed with AT130 Reveal Secondary and Tertiary Structural Changes and Mechanisms of Allostery". In: *Viruses* 13.4 (2021). ISSN: 1999-4915.

[45] E. D. I. M. O. R. N. P. R. B. A. B. E. W. R. C.-B. E. U. W. N. A. R. R. T. Peter G. Stockley; Simon J. White and R. Twarock. "Bacteriophage MS2 genomic RNA encodes an assembly instruction manual for its capsid". In: *Bacteriophage* 6.1 (2016). PMID: 27144089, e1157666. eprint: `https://doi.org/10.1080/21597081.2016.1157666`.

[46] C. A. Piedade, A. E. N. Ferreira, and C. Cordeiro. "How to disassemble a virus capsid a computational approach". In: vol. 3. Cited by: 1; All Open Access, Green Open Access, Hybrid Gold Open Access. 2017, pp. 217–222.

[47] Ó. Rolfsson et al. "Direct Evidence for Packaging Signal-Mediated Assembly of Bacteriophage MS2". In: *Journal of Molecular Biology* 428.2, Part B (2016), pp. 431–448. ISSN: 0022-2836.

[48] S. Singh and A. Zlotnick. "Observed Hysteresis of Virus Capsid Disassembly Is Implicit in Kinetic Models of Assembly*". In: *Journal of Biological Chemistry* 278.20 (2003), pp. 18249–18255. ISSN: 0021-9258.

[49] G. R. Smith, L. Xie, B. Lee, and R. Schwartz. "Applying Molecular Crowding Models to Simulations of Virus Capsid Assembly In Vitro". In: *Biophysical Journal* 106.1 (Jan. 2014), pp. 310–320. ISSN: 0006-3495.

[50] R. Tarjan. "Depth-First Search and Linear Graph Algorithms". In: *SIAM Journal on Computing* 1.2 (1972), pp. 146–160. eprint: `https://doi.org/10.1137/0201010`.

[51] C. Thèves, E. Crubézy, and P. Biagini. "History of Smallpox and Its Spread in Human Populations". In: *Microbiology Spectrum* 4.4 (2016). eprint: `https://journals.asm.org/doi/pdf/10.1128/microbiolspec.poh-0004-2014`.

[52] K. Toropova, P. G. Stockley, and N. A. Ranson. "Visualising a Viral RNA Genome Poised for Release from Its Receptor Complex". In: *Journal of Molecular Biology* 408.3 (2011), pp. 408–419. ISSN: 0022-2836.

[53] S. TRIPATHI, J. Y. SUZUKI, S. A. FERREIRA, and D. GONSALVES. "Papaya ringspot virus-P: characteristics, pathogenicity, sequence variability and control". In: *Molecular Plant Pathology* 9.3 (2008), pp. 269–280. eprint: https://bsppjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1364-3703.2008.00467.x.

[54] R. Twarock and A. Luque. "Structural puzzles in virology solved with an overarching icosahedral design principle". In: *Nature Communications* 10.1 (Sept. 2019), p. 4414. ISSN: 2041-1723.

[55] UNAIDs. *Global HIV & AIDS statistics — Fact sheet 2023*. URL: https://www.unaids.org/en/resources/fact-sheet. (accessed: 03/06/2024).

[56] E. A. B. Week. *Kenya: Disease Hits Kenya Maize Expectations*. URL: https://allafrica.com/stories/201503101043.html. (accessed: 04/07/2024).

[57] W.-S. Wei, A. Trubiano, C. Sigl, S. Paquay, H. Dietz, M. Hagan, and S. Fraden. "Hierarchical assembly is more robust than egalitarian assembly in synthetic capsids". In: *Proceedings of the National Academy of Sciences of the United States of America* 121 (Feb. 2024), e2312775121.

[58] E. Wroblewski et al. "Visualizing Viral RNA Packaging Signals in Action". In: *Journal of Molecular Biology* 436.22 (2024), p. 168765. ISSN: 0022-2836.

[59] A. Zlotnick. "To Build a Virus Capsid: An Equilibrium Model of the Self Assembly of Polyhedral Protein Complexes". In: *Journal of Molecular Biology* 241.1 (1994), pp. 59–67. ISSN: 0022-2836.