Force-Aware Self-Assembly of Bridges by a Swarm of Autonomous Robots

A Thesis Submitted for the Degree of Doctor of Philosophy

Edward Bray

Supervised by Dr Roderich Groß and
Dr Dana Damian

Department of Automatic Control and Systems Engineering

The University of Sheffield

June 27, 2023

Abstract

Self-assembling modular robotic systems comprise multiple agents which form connections with each other to create different arrangements to suit their current task. Existing control algorithms for these systems typically do not consider the forces in the connections between agents, potentially leading to damage to the robots due to unacceptably high forces. This thesis considers how force-aware methods can be used to guide the agents into self-assembling structures that will not break. As an example scenario where force-aware control shows great potential, the self-assembly of a bridge across a gap in the terrain is considered. This would enable a group of robots to explore otherwise unreachable environments, but requires careful consideration to ensure the bridge does not collapse under its own weight. This process is split into three stages, and separate algorithms are developed to allow each stage to be completed: each algorithm runs in a distributed manner across each agent, and responds to measurements of forces made by the agents to produce strong structures that are not designed in advance by a human. A cantilever is first constructed from one side of the gap until the other is reached, whereupon the structure is optimised by removing agents that are no longer providing support. Finally, the bridge is deconstructed when it is no longer required. The algorithms are first verified in simulation, then a novel hardware platform is developed to demonstrate their applicability in real life. The cantilevers and bridges built are found to be close to the optimal with respect to the number of agents required to build a stable structure of a given length. The forces within the structures can be controlled by choosing suitable allowable limits. This work hopes to inspire future researchers to explore how force-aware methods can be applied to other problems in self-assembling modular robotics.

Acknowledgements

This thesis represents several years of my working life, and would not have been possible without the help of a large number of people. I would like to begin by thanking my supervisor, Dr Roderich Groß. His guidance and comments have been invaluable throughout the process. He has taught me how become a researcher, and to look at my own work with a critical eye. Thanks also to my second supervisor, Dr Dana Damian, for her support when needed.

I'd also like to thank the numerous technical and support staff for their help with the hardware work. Garry Turner in the Sheffield Robotics Lab has been of great assistance during the physical experiments. Manufacturing the hardware would not have been possible without the help of both the ACSE financial team and technicians.

While developing the updated HyMod platform, it was incredibly helpful to talk with the original designer, Dr Chris Parrott. He answered many questions about the details of his work. Without this assistance, I'm sure the hardware development would have taken significantly longer.

Despite the pandemic, I have always preferred to work from the office instead of my house. This is in no small part thanks to the great people I interact with on a daily basis. I am very grateful to the past and present members of B12 and the Natural Robotics Lab for their discussions, both about work and more important things!

Work-life balance is of great importance to me, and I have been very lucky to be able to carry out this work in Sheffield, with the Peak District on the doorstep. For whatever reason, my method of clearing my mind after a hard day has been to go outside and get scared clinging to a rockface or hurtling down a rocky hill on a bike. I am eternally grateful to my friends, near and far, who have cheered me on, held my ropes, and generally explored the natural playground with me.

Doing a PhD in the first place was not something that I've always aspired to. It was only when I could take time after my degree to think about what I really wanted in life that I chose this path... for better or worse! I will always be thankful to my parents for raising me with a critical eye and letting me choose my own path. They have always supported my decisions, even when that involved spending many months volunteering at Kandersteg International Scout Centre instead of getting a real job!

I, finalment, gràcies a la meva parella, Ari. Perquè la vida és més que la robòtica i l' escalada.

Contents

Chapter 1

Introduction

Throughout history, humans have relied on technological innovations to improve their quality of life. From ancient mechanisms such as the Archimedes Screw, through the boom of steam powered machines during the Industrial Revolution, to the mass availability of electric devices seen today, engineering advancements have made many laborious tasks easier. Robotics is one such technology that has shown great potential to be applied to a wide range of scenarios. In the last century, robots have become ubiquitous in settings such as factories and warehouses, where they perform repetitive tasks in well-defined environments efficiently around the clock. They are also increasingly being deployed outside of these controlled arenas, with applications such as medical devices and self-driving cars showing the potential of robotics in more complex domains. With such diverse possible applications, there is a great demand for robust and capable robotic hardware, controlled by safe and reliable algorithms.

Humans are capable of great things individually, but can work together to achieve even more. The same is true for robots, who can collaborate to form a *multi-robot system* (MRS) which has several advantages [1]. An MRS could be more suitable for tasks too complex for a single robot to perform. When multiple tasks can be completed in parallel, an MRS is able to finish all of them faster than a single robot tackling them sequentially; MRSs are particularly advantageous when these tasks are distributed across a large area, as the time spent moving between the task locations is reduced. Using one robot could also leave the system vulnerable to a single point of failure, increasing the risk of the task not being completed: MRSs are potentially more robust through increased redundancy. Designing a single robot with a wide range of capabilities is often more challenging than several robots that are more specialised. Finally, using multiple robots increases the flexibility of the system, as the number of robots deployed and their precise capabilities can be varied depending on the task at hand.

A popular branch of MRS research is *modular reconfigurable robotics* [2, 3]. This approach considers systems comprising multiple relatively simple robotic modules, which form physical connections between each other and thus create different structures based on the current task. These systems are therefore highly versatile, scalable, and typically robust to the failures of individual agents. Their modules are either arranged into a particular design by a human or are able to do so themselves in a process called *self-assembly* [4]. The work in this thesis considers such *self-assembling modular robots*. They have a wide range of possible applications, as they could be arranged to create configurations suitable to tackle virtually any problem that can currently be solved by single-robot systems, while also sharing many of the same advantages of MRSs.

One area in which self-assembling modular robots show great potential is in all-terrain navigation. Robots that could autonomously convert between different locomotion modes depending on the scenario could be very beneficial in moving over unpredictable and varied terrain: for example, a car configuration could drive quickly around flat areas, while a snake shape could fit through small gaps or pipes. This could be particularly advantageous while searching for survivors underneath the rubble of collapsed buildings [5]. MRSs are attractive for this application in general, as they can spread out over a large search area; this area may be hazardous and so cause damage to a number of agents, so the redundancy of MRSs is also beneficial. Deploying self-assembling modular robots could be particularly advantageous as they would be able to configure themselves differently to suit the wide range of terrains encountered.

Further applications of self-assembling modular robotics are almost limitless. The versatility of manufacturing facilities could be increased by replacing existing specialist robotic manipulators with self-assembling modular systems capable of a wider range of functionalities [6]. Self-assembling robotic furniture would allow residents of small properties to get the most out of their limited living space, and similar benefits could be seen by companies applying the same technology to their offices [7]. Self-assembling modular robots also show great potential in space applications, as their versatility would minimise the total payload required to be sent into space to accomplish a number of tasks; they could be applied both to routine operations involving satellites around Earth, or more futuristic missions to other worlds [8].

In addition to their wide range of applications, self-assembling modular robots are also attractive as they could promote sustainable practices in robotics, something that is of paramount importance in all aspects of life [9]. The versatile nature of modular robotic platforms reduces the consumption of resources when manufacturing robots to perform different functions. Modular robotic systems also allow for faulty parts to be easily replaced without discarding the whole system, further reducing waste. By introducing self-assembly into these systems, their ease of use is increased, making the technology more accessible to users without specialist knowledge.

1.1 Motivation

Despite the huge potential of self-assembling modular robotic systems outlined above, they are rarely used outside of the laboratory [10]. Many factors contribute to this, including the cost and availability of suitable robotic platforms, which are typically prototype systems with low technical readiness levels. Another important aspect is that the control algorithms are often developed in idealised simulations that do not consider every salient real-world limitation. This is understandable, as it allows researchers to focus on the details of specific problems relating to this challenging field. However, it potentially increases the challenges in modifying the resulting control algorithms to the real world. There is therefore a great need for the development of algorithms that account for a wide range of real-world phenomena to ease the transition from simulated to real systems.

One particularly important limitation of real-world self-assembling modular robots is that the connections between agents have a finite strength, so will break if forces within them get too high. This could cause damage to the modules, potentially leaving them unable to complete their task. Despite this, many studies assume that these links are sufficiently strong that they will not break under normal use. The deployment of self-assembling modular robots in the real world would therefore greatly benefit from acknowledging this limitation and developing algorithms to account for it. If robots could detect the current or expected loading, they could autonomously self-assemble into different shapes to ensure this load could be supported, while not wasting modules building structures that are stronger than necessary. Such *force-aware* methods of control would make them significantly more versatile and applicable to real-world problems.

Using measurements of force to influence the control of robotic systems is a widely employed method in certain applications, most commonly the control of robotic manipulators [11]. Attempting to complete tasks by specifying positions to move a manipulator into can lead to unacceptable forces being generated either within the robot itself or between the robot and its environment. By utilising force-aware control schemes, a wide range of functions can be achieved without causing damage to the robot or objects within its environment. Robotic manipulators using such methods can therefore operate in less well-structured scenarios than those relying on pure position control, and can also collaborate with humans in a safer manner.

Construction is one very promising application of robotics, and MRSs in particular, that lends itself to force-aware control. Construction sites are potentially hazardous environments, containing multiple tasks that can be completed independently of each other, so therefore the parallel and robust nature in which MRSs operate is advantageous [12]. The construction of structures by an MRS could be achieved by having a human design a structure which the robots find a suitable method to build [13]. By incorporating force-aware methods, safe assembly sequences can be found that ensure forces within the structure are kept at acceptable levels throughout construction [14].

Having robots build structures designed by humans requires different arrangements to be defined depending on the situation. A more adaptive approach is to specify a high-level task which the robots complete by determining both a suitable structure and its construction sequence themselves. Force-aware control is highly suited to this scenario, as it can be used to ensure the structure is strong enough to allow the robots to complete their specific task. Funes and Pollack showed how computers can produce strong structures through an evolutionary process, which are also verified to be stable when constructed by hand in real-life [15]. This work was expanded by Brodbeck and Iida, who demonstrated how such structures could be both autonomously designed and constructed by a robotic manipulator [16]. A similar approach has also been demonstrated with MRSs, illustrating how a group of robots can build a structure together while incorporating local force measurements to ensure it does not collapse [17].

Incorporating force-aware control methods into self-assembling modular robots could benefit the system in many ways. Suzuki *et al.* have shown how a group of self-assembling robots can move together to achieve collective motion, while including measurements of force within the connectors to prevent them from breaking [18]. Robotic manipulators formed from self-assembling modular robots could incorporate force measurements to ensure they are suitable for the task at hand, potentially recruiting additional modules to match the strength of the machine to the current loading. The same idea could be applied to modules arranged to move in a walking gait: the number and strength of the legs could vary depending on any load that the modules are transporting, or the nature of the terrain they are moving across.

An area in which force-aware control could be very beneficial to self-assembling modular robots is in constructing bridges to enable the group of robots to cross a gap in the terrain several bodylengths wide. One method of achieving this task could be for each agent to jump, but this requires a relatively large amount of energy per agent and is hard to perform accurately, especially if robots must coordinate to avoid mid-air collisions [19]. Constructing a bridge that can be used by agents to cross the gap would therefore potentially be more suitable. Such a bridge could be built from external building materials: while this could be cost-effective, it either relies on specialist materials which must be delivered to the construction site [20], or for there to be suitable materials nearby [21]. Self-assembling the bridge from the robots themselves means the building material effectively transports itself to the structure, increasing the range of environments the approach could be applied to. The robots could also autonomously reconfigure the structure based on the current requirements and deconstruct it when it is no longer required, further increasing the efficiency of this approach.

Force-aware methods of self-assembling bridges are currently not widely researched, despite the potential benefits of this approach. In 2000, Inou *et al.* performed some of the first work in this area, showing how a group of force-aware robots could self-assemble into a bridge to transport a load from one side of a gap to another [22]. Their approach demonstrated the potential of force-aware control to adaptively self-assemble structures depending on the current loading. However, it only allowed a single agent to add to the structure at a time, resulting in long construction times. Furthermore, a large number of messages are required to be reliably passed between agents, reducing both the robustness of the approach to the failure of modules, and its scalability. The authors continued this work by considering how the resulting bridges could be deconstructed when the load has passed to the other side of the gap [23]. Safe deconstruction is an important aspect of self-assembled structures, however the described method occasionally results in an unresolved deadlock, where agents repeatedly add and remove from the same locations and thus progress towards task completion stalls. In both works, the algorithms were validated in limited simulations, and not compared to any baselines, such as the minimum number of agents required to span a gap of a given length without structural failure.

More recently, the *ReactiveBuild* algorithm has been proposed to bring about force-aware self-assembly of a variety of static structures, including bridges and towers [24]. In this approach, robots move towards a goal location, and are able to call adjacent agents to stop and provide reinforcement when they sense their links are close to breaking. This is a promising method, but can result in the creation of large support structures, which slows the growth of the structure towards the goal, thus preventing it from being reached with a limited number of modules. As in the work of Inou *et al.*, construction takes a long time as agents are added to the structure one by one.

Force-aware self-assembly shows great potential, though further research must be performed to realise these benefits in a robust manner and on real-world systems. While previous works have shown how structures can be self-assembled without collapse, they only allow agents to add themselves to the structure one at a time, therefore making the self-assembly process slower than if multiple agents could build on the structure at once. Research into how structures could be modified in response to changing scenarios has not been considered, and the problem of how structures can be safely deconstructed without deadlock remains unsolved. Furthermore, while platforms capable of measuring force in connectors have been developed, force-aware self-assembly has only been shown in simulation, not on real-life robotic systems.

1.2 Problem Definition

This thesis examines how self-assembling modular robots can use a force-aware approach to self-assembly to exploit the possible advantages explored above. As an example scenario where force-aware methods would be beneficial, the problem of how a group of robots can cross a gap in the terrain is considered. This scenario might be encountered by such a group when exploring unstructured environments, such as when performing search and rescue operations after a natural disaster.

The specific problem considered in this thesis is as follows. A group of self-assembling modular robots are required to complete a task on the other side of a gap in their terrain. To reach this task, they must self-assemble a bridge across the gap, which can be crossed by other agents. The first stage of this is *cantilever construction*, in which the agents self-assemble a structure connected to only one side of the gap that extends towards the opposite side. When the other side is reached, *bridge optimisation* can occur, in which agents within the structure that are no longer providing support are removed so that they can work on completing the main task. Eventually, the bridge will no longer be required as either all the agents have crossed the gap, or the task is complete and they have all returned to the original side. At this point, *bridge deconstruction* begins: agents are first added to the structure to reinforce it so that it will not collapse when it is disconnected from one side of the gap, then the bridge is deconstructed. At all stages during the construction, optimisation, and deconstruction, the agents are guided by measurements of forces within the structure to ensure that links between agents do not break and cause the structure to collapse. A human is not required to design specific structures to create bridges of a given length or strength, but rather these designs emerge from agents responding to local measurements of forces within the structures.

1.3 Aims and Objectives

Given the problem definition above, the overall aim of this thesis is to develop algorithms that will enable self-assembling modular robots to form bridges. These bridges shall not be preplanned, but should be strong enough such that they will not collapse during the self-assembly process or while in use.

The specific objectives that will be met to fulfil this aim are:

- To create a simulation environment that will enable the development of control algorithms for self-assembling modular robots that is able to rapidly and accurately model and calculate the forces between modules in the structure. This environment shall be restricted to a 2D grid for simplicity.

- To develop novel algorithms to enable the self-assembling modular robots to form bridges across a gap without collapsing. These algorithms should first allow the structure to be built from one side until the other is reached. When this occurs, the additional support will enable a second stage where the number of agents in the structure is safely reduced so they can complete other tasks. Finally, when the structure is no longer required, it should be able to be safely dismantled.

- To calculate optimal structures against which to compare the performance of the algorithms. A cantilever of N agents is considered optimal if it safely spans the maximum distance possible for this N. Optimal bridges are defined as those that safely span a given length with the minimum number of agents.

- To verify and analyse the performance of these algorithms with a large number of robots in simulation.

- To develop suitable novel robotic hardware to deploy these algorithms on in real-life. As in the simulations, this hardware shall be designed only to create 2D, grid-based structures.

- To demonstrate the real-world applicability of these algorithms through deployment on a physical robotic platform.

1.4 Preview of Contributions

The contributions of this thesis are as follows:

- A novel distributed algorithm to allow the force-aware construction of cantilevers from self-assembling modular robots that add themselves to the cantilever one at a time. The algorithm incorporates local force measurements to ensure the structure does not collapse during construction. It does not require a human to design the structure: agents instead use the measured force information to produce a suitable shape. Two variants are developed, one in which agents within the structure coordinate to inform the agent that is currently placing of the maximum forces in each column, and another where this agent only receives the force measurements in links connected to agents on the perimeter of the structure. Simulation studies with large numbers of agents show that both variants build cantilevers close to the precomputed optimal length for a given number of agents for different link strengths; performance is not significantly diminished by reducing the amount of information passed to the currently-placing agent.

- An extension of the aforementioned algorithm to allow multiple agents to move around the structure simultaneously as they self-assemble cantilevers. This algorithm also contains behaviours to resolve collisions between agents travelling in opposite directions to ensure the continual construction of the cantilever. It is validated in simulation, and shown to construct cantilevers of comparable length and strength to the previous algorithm, but significantly faster.

- A further distributed and force-aware self-assembly algorithm that enables the number of agents within the bridge to be reduced when the cantilever reaches the other side of the gap. This algorithm optimises the structure by removing and repositioning agents while ensuring it is still able to support itself. It allows for multiple agents to move around the structure at once, and does not produce designs preplanned by a human. Bridges resulting from the previous cantilever construction algorithm are used as starting configurations in simulation to verify that the algorithm can reduce the number of agents in bridges to almost the precomputed optimal number in a wide range of scenarios.

- An additional algorithm to deconstruct these bridges when they are no longer required. This algorithm is again distributed and force-aware to ensure that it is scalable and builds structures that will not collapse; multiple agents can move around the structure at a time and preplanned structures are not built. Elastic beam theory is used to enable agents to approximate what the force distribution in the structure would be if one of the supports were disconnected. This algorithm is also verified in simulation for a range of scenarios. It is shown to be able to reliably deconstruct all bridges tested, requiring a similar number of agents than were used by the initial cantilever construction algorithm to span the gap. The maximum forces within connections during the operation of this algorithm are found to be comparable to those during the initial construction of this bridge, and more than during the subsequent optimisation stage.

- The design and manufacture of a novel hardware platform on which to deploy these algorithms in the real world. This is based on the existing HyMod system [25], but it is extensively modified to incorporate the specific features necessary for these algorithms. In particular, novel force-aware versions of the HiGen connector [26] are developed to allow agents to measure the forces within their links to determine how close they are to failure. Two modules are designed: a passive module to allow the verification of the algorithms in real-life, and an active metamodule to demonstrate how modules would move around the structure in a fully autonomous system. Ten passive modules and one active metamodule are manufactured and characterised.

- Verification of the algorithms in real life, demonstrating how they provide a viable method of using self-assembling modular robots to create bridges while ensuring structures are able to support their own weight. Modules are moved around the structure by hand, and decide for themselves how the structure should be built based on the force information received, following the developed algorithms. The performance in real-life is evaluated and similar trends are observed to those seen in simulation.

1.5 Publications

This thesis contains original contributions the author has made to scientific knowledge. The work presented in this thesis has so far led to two peer-reviewed publications:

- **E. Bray** and R. Groß, “Distributed Self-Assembly of Cantilevers by Force-Aware Robots,” in *International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2021, pp. 110–118 (winner of conference’s *Best Student Paper* award)

- **E. Bray** and R. Groß, "Distributed Optimisation and Deconstruction of Bridges by Self-Assembling Robots," in *Robotics: Science and System*, 2022

These publications were presented orally by the author at their respective conferences: these were held in Cambridge (UK) and New York (USA).

1.6 Thesis Outline

The remainder of this thesis is organised as follows:

- Chapter 2 contains related works and a discussion of the field of self-assembling modular robots to date. Section 2.1 begins the chapter by introducing inspiration from biological systems. Current hardware platforms are described in Section 2.2, before an overview of existing algorithms is given in Section 2.3. Existing force-aware self-assembly algorithms are then discussed in detail in Section 2.4. Section 2.5 summarises and concludes the chapter.

- Chapter 3 presents the cantilever construction stage of the self-assembly of bridges. Section 3.1 formally defines the problem, then the simulation environment used to evaluate the algorithms is presented in Section 3.2. Optimal cantilevers are computed in Section 3.3 to compare the performance of the self-assembly algorithms against. These algorithms are presented in Section 3.4: two variants of an algorithm in which agents place one at a time are first described, then another algorithm in which multiple agents can move around the structure at once is given. Simulation studies are performed in Section 3.5 to demonstrate the performance of the algorithms for large numbers of agents. Section 3.6 summarises and concludes the chapter.

- Chapter 4 presents the bridge optimisation stage of the self-assembly of bridges. Section 4.1 formally defines the problem, then optimal bridges are computed in Section 4.3 to compare the performance of the self-assembly algorithm against. The bridge optimisation algorithm is presented in Section 4.4, and validated extensively in simulation in Section 4.5. Section 4.6 summarises and concludes the chapter.

- Chapter 5 presents the bridge deconstruction stage of the self-assembly of bridges. Section 5.1 formally defines the problem. The bridge deconstruction algorithm is presented in Section 5.2, including a description of the method by which forces in the bridge if it were to be released from one side are calculated. The algorithm is validated extensively in simulation in Section 5.3. Section 5.4 summarises and concludes the chapter.

- Chapter 6 presents a validation of the algorithms with real-life hardware. The design of the hardware is shown in Section 6.1, and it is characterised in Section 6.2. The algorithms are slightly modified to deploy them on this hardware, as described in Section 6.3. Thorough experiments are performed and analysed in Section 6.4. Section 6.5 summarises and concludes the chapter.

- Chapter 7 summarises the thesis, and discusses the conclusions that can be drawn. Potential avenues for future work are identified and discussed in Section 7.1.

Chapter 2

Background

This chapter reviews the existing literature related the work presented in this thesis to provide context for the novel developments contained herein. Firstly, Section 2.1 introduces a number of studies from the field of biology that have been inspirational in conceiving the methods explored by researchers developing self-assembling modular robotic systems. Section 2.2 describes the common challenges faced when designing physical self-assembling modular robots, and provides an overview of how they have been addressed by engineers to date. A range of approaches explored by researchers to control the self-assembly of these robotic platforms are presented in Section 2.3. This thesis specifically considers force-aware self-assembly, so existing works demonstrating this approach are explored in depth in Section 2.4. The chapter is concluded in Section 2.5.

2.1 Inspiration from Biological Systems

Roboticists are often inspired by nature when designing robotic systems. Over millions of years, evolution has resulted in both bodies that are capable of sophisticated and graceful movements, and brains that help animals to thrive and adapt to different environments. In many species, animals work together, displaying relatively simple behaviours such as flocking [27], or complex coordination such as generating large waves to wash prey into the ocean [28].

At a high level, this thesis considers how structures can be built by a team of robots. Groups of animals collaborating to build structures is observed in a wide range of species. Beavers build dams and lodges in rivers to ensure their safety and food supply [29], while birds construct nests in which to raise their young [30]. Perhaps some of the most impressive examples of animals working together to build structures come from termites, who work together to build nests several metres tall [31]. Despite the limited intelligence of each individual termite, they are able to follow simple rules based on indirect communication with other agents through cues left in the environment. This form of communication is called *stigmergy*, and has been proposed as the mechanism by which numerous species display complex emergent behaviours [32].

The structures built by animals that are most relevant to the work contained within this thesis are the self-assembled structures built by ants. Large numbers of ants are able to self-assemble into different temporary structures to help the colony thrive, such as rafts [33], bivouacs [34], towers [35], and pothole plugs [36]. Ants have a very high strength-

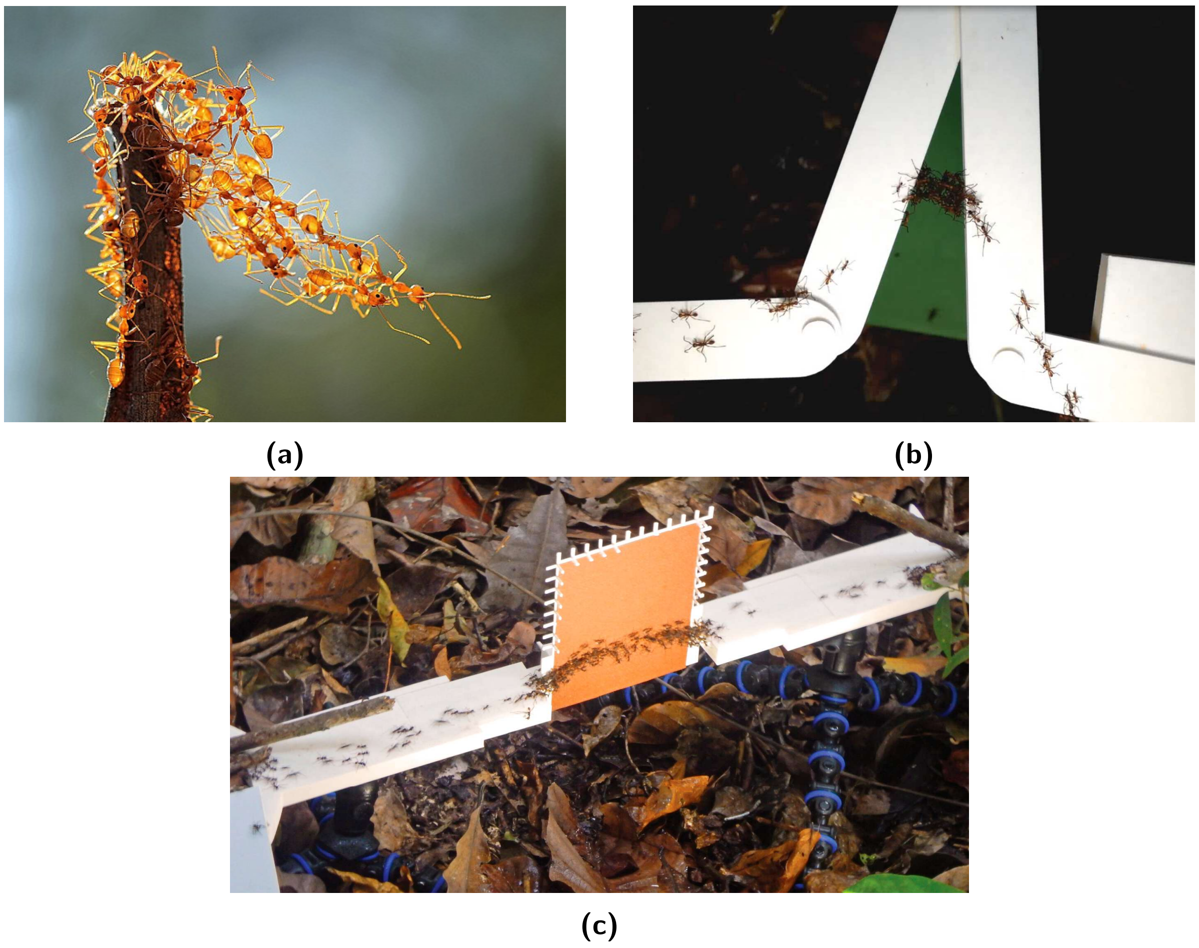


Figure 2.1. Examples of bridges built by ants. **(a)** A cantilever extending from one support to enable construction of a bridge to a new location, © 2014 Adhi Prayoga / Solent News. **(b)** A bridge forming a shortcut between two locations that have previously been explored. **(c)** A scaffold built against an inclined surface to stop agents from slipping down it. Reprinted from [38–40] respectively.

to-weight ratio and are able to attach to each other at arbitrary locations [37], enabling them to self-assemble into complex shapes: such properties are also very desirable in self-assembling robots.

The main source of inspiration for this thesis comes from how ants self-assemble bridges out of their bodies, as in the examples shown in Figure 2.1. The African weaver ant *Oecophylla longinoda* has been observed to build bridges by initially extending a cantilever from one side of a gap until the other is reached [41] (Figure 2.1a). These bridges allow the colony to reach areas that would otherwise be unreachable, helping them to forage for new resources. Their construction involves large numbers of ants accumulating on one side of the gap, who then start to reach out across it. As more ants arrive, a large ball of ants builds on this support. This acts as a buttress to transfer the load onto the support, enabling the structure to extend further without collapsing, until eventually the other side of the gap is reached.

The army ant *Eciton hamatum* self-assembles bridges which act as shortcuts between points that the colony has already explored (Figure 2.1b). These bridges adapt to the traffic flow over the bridge, and exhibit a cost-benefit trade-off: longer structures will require more agents, thus reducing the number that can forage, but allow the remaining foraging agents to do so more efficiently [39]. *Eciton burchellii* can self-assemble scaffolds between two points separated by an inclined plane which ants would otherwise slip on [40] (Figure 2.1c). While these are not true bridges, these structures demonstrate another way in which ants use self-assembly to navigate their environment in a more efficient manner. Similar behaviours implemented in robotic systems would enable them to reliably explore unpredictable real-world environments.

2.2 Self-Assembling Modular Robotic Systems

The physical design of self-assembling modular robots has been an active research topic for decades. Since Fukuda and Nakagawa launched research into modular reconfigurable robots with the Dynamically Reconfigurable Robotic System in 1987 [42], researchers have designed a wide range of robotic platforms capable of self-assembly. All of these platforms should consider four common challenges:

1. **Topology:** how the self-assembled agents should be arranged with respect to one another.
2. **Connections:** how the physical connections between agents should be made.
3. **Actuation:** how the agents should move, both around their environment and with relation to one another.
4. **Communications:** how the agents should communicate with other agents, a human operator, or both.

Each of these challenges is considered separately below, and a range of promising solutions developed to date are presented and discussed. Platforms typically consist of only a single type of module, thus are called *homogeneous*. A small number of systems comprise multiple different types of module, so are referred to as *heterogeneous* platforms.

2.2.1 System Topologies

The topology of the platform describes how the agents connect to each other. This affects the possible structures that can be built, and so heavily influences the functionality of the collective. Systems are traditionally described as *chain-type*, *lattice-type*, or *mobile* [43]. However, recently a fourth topology has risen in popularity, called *freeform*. The advantages and disadvantages of each class are described below, along with some examples of robotic platforms that fit within them. Some systems exhibit traits of a number of topologies, so are often referred to as *hybrid-type*: such systems are mentioned here in the category that they most-resemble.

2.2.1.A Chain-Type

In chain-type modular robots, agents attach to each other end-to-end, occasionally employing additional modules to allow separate branches to be made. Each module typically contains one or more degrees of freedom, and is able to move around the environment

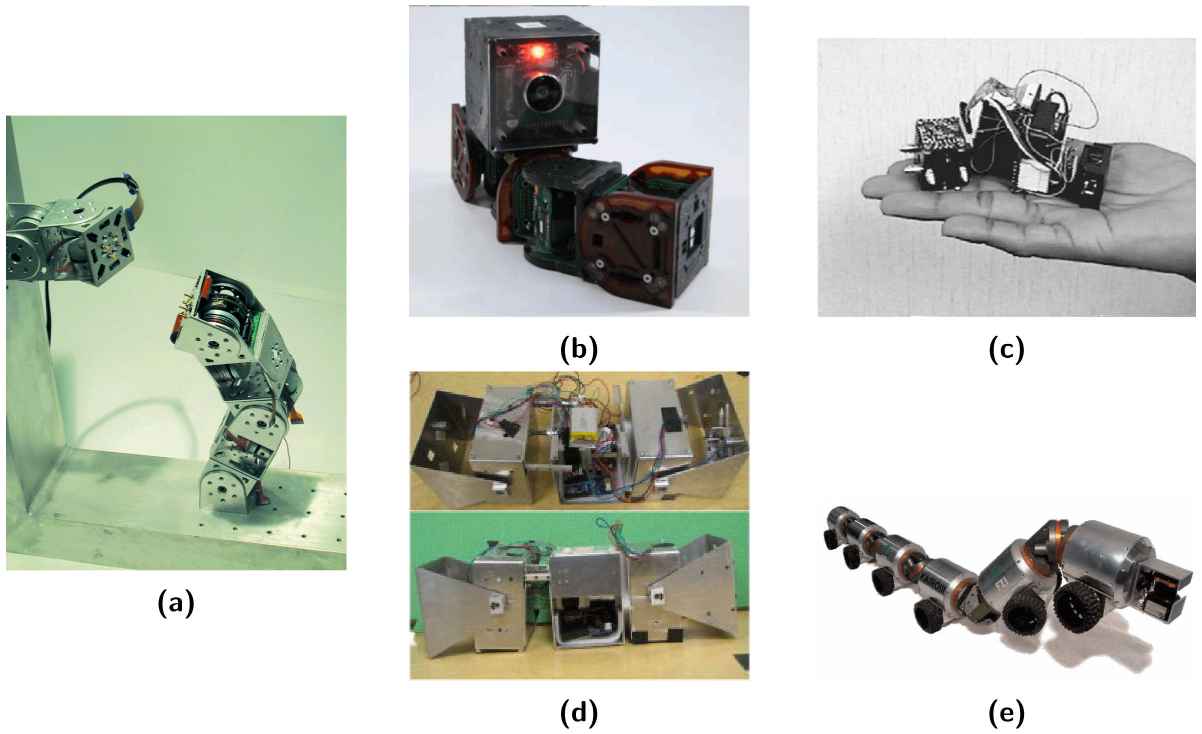


Figure 2.2. Examples of chain-type self-assembling modular robotic systems: **(a)** PolyBot G3 © 2007 IEEE, **(b)** CKBot © 2007 IEEE, **(c)** CONRO © 2002 IEEE, **(d)**, ModRED[1], and **(e)** KAIRO 3 © 2014 IEEE. Reprinted from [43, 46–49] respectively.

on their own. The high mobility of agents means that they must go through careful alignment procedures while docking, but also leads to highly-manoeuvrable self-assembled structures. These structures can move around with a variety of gaits, including snake-like crawling [44] or rolling locomotion [45].

One of the earliest chain-type modular robotic platforms is Polypod [50]. This heterogeneous platform comprises two types of module, called *segments* and *nodes*. Segments are nominally cubic with two connectors on opposite faces, and are capable of both prismatic and rotational motion of the connectors relative to each other. Nodes are passive cubes that contain six connectors to allow branches in the chain. These modules can be arranged in several ways to demonstrate different methods of locomotion, including walking and caterpillar gaits.

The design of Polypod inspired the authors to develop PolyBot [51] (Figure 2.2a). This platform is homogeneous, and also comprises robots that are roughly cubic with two connectors on opposite faces. However, the prismatic actuation is removed to allow the modules to rotate through a greater range, and to create space for an active connection mechanism that allows modules to autonomously control their connections to each other. The motor that actuates the module rotation protruded from the module body in early versions, but later generations refined the design to fit all the components within a 50 mm

[1]Reprinted with permission from J. Baca, S. G. M. Hossain, P. Dasgupta, C. A. Nelson, and A. Dutta, "ModRED: Hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration," *Robotics and Autonomous Systems*, Reconfigurable Modular Robotics, vol. 62, no. 7, pp. 1002–1015, 2014

cube [52]. The CKBot platform builds on PolyBot by incorporating the same kinematic design, but with simplified hardware to make modules less expensive to manufacture [53] (Figure 2.2b). Additional modules, such as cameras, can be added to introduce additional functionality to the system [46].

The CONRO system incorporates a very different module morphology to the previous systems [47] (Figure 2.2c). These modules comprise two connectors, separated from each other by the robot body. One connector has connection surfaces on three faces, giving a total of four possible connections for each module to allow for branching structures. The body contains two motors to vary the yaw and pitch of the connectors relative to each other. This platform is able to build highly mobile structures, including snakes or hexapods.

The above examples all date from the around the turn of the millennium. In the years since, this topology has fallen out of favour with the research community, though systems such as ModRED show how novel chain-type systems are still being developed [48] (Figure 2.2d). These modules contain two opposing connectors similar to previous systems, but each connector has its own rotational degree of freedom, and each module also has a prismatic joint between the connectors. The reasons for this reduction in popularity are unclear: it could be that the downsides associated with controlling numerous degrees of freedom have made chain-type topologies less attractive, or perhaps the versatility offered by other topologies makes them more attractive. Regardless, chain-type systems still have an important niche in exploring narrow passageways, for example in pipes. Robots such as KAIRO [49] (Figure 2.2e) show chain-type modular robotic systems can have important real-world applications, especially within this niche [54].

2.2.1.B Lattice-Type

The agents in lattice-type self-assembling modular robotic platforms connect to each other in regular locations within a lattice. The lattice is typically square or cubic, but some systems incorporate hexagonal [55] or triangular [56] arrangements. The regular arrangement of modules means alignment of connectors is simplified compared to chain-type systems, making self-assembly easier. Purely lattice-type modular robots are not able to independently move outside the lattice in a controllable manner. Example systems are grouped below by the type of lattice they form, a classification inspired by Parrott [57].

The first type of lattice is the *translational* lattice. In these systems, modules contain only prismatic joints. They usually reside in a square 2D grid, and thus can only move into locations in their von Neumann neighbourhood. Two methods of achieving this motion have been demonstrated. One approach is for the faces of modules to extend in order to push them around the lattice, as demonstrated by Crystalline [64] (Figure 2.3a) and Telecubes [65]. The alternative approach is to slide along the surface of adjacent modules: Pamecha *et al.* [66] and the CHOBIE II platform [67] (Figure 2.3b) achieve this by incorporating tracks within the faces of modules which others can travel along using wheels, while EM-Cubes use magnets [68].

Systems designed with a *rotational* lattice arrangement incorporate rotational joints, allowing them to move to a greater range of locations. Fracta is an early example of such a system, consisting of hexagonal modules that move in a 2D plane by pivoting about their corners [55]. There are also several cubic systems that pivot around their edges, such as 3D M-Blocks [60] (Figure 2.3c), Kubits [69], and ElectroVoxels [70]. Similarly, 3D Catoms

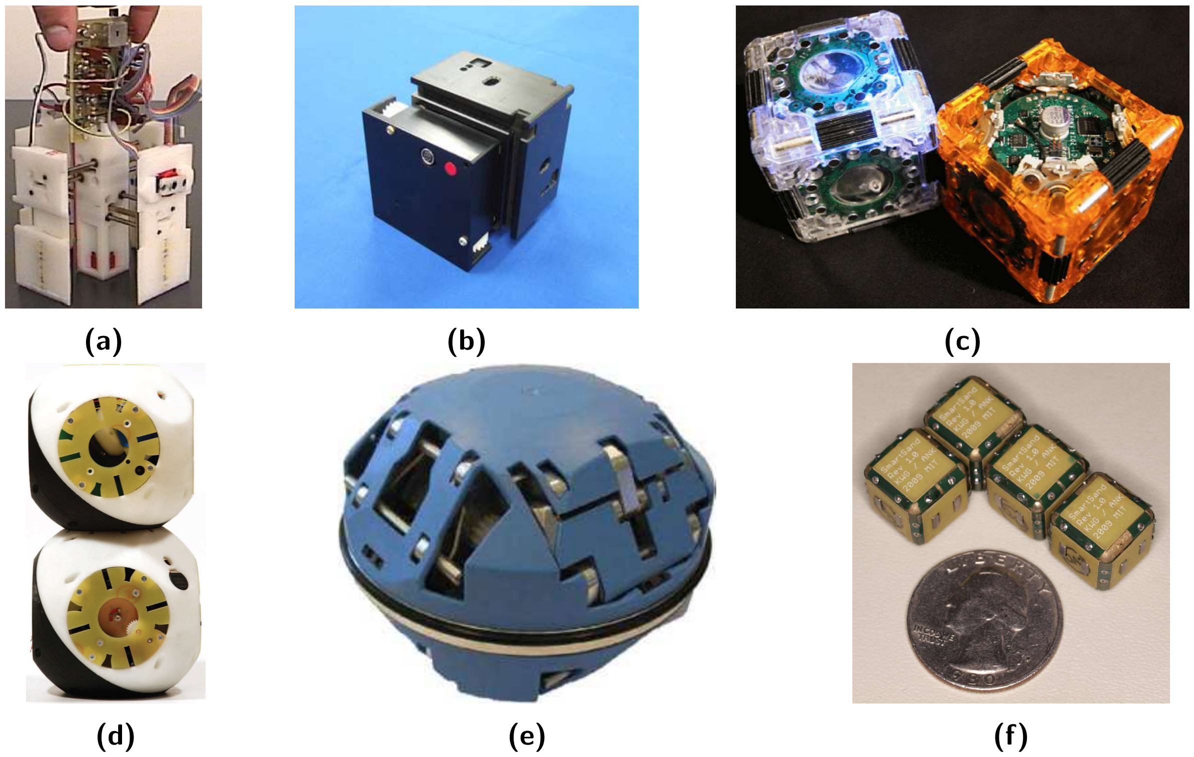


Figure 2.3. Examples of lattice-type self-assembling modular robotic systems: **(a)** Crystalline © 2000 IEEE, **(b)** CHOBIE II © 2008 IEEE **(c)** 3D M-Blocks © 2015 IEEE, **(d)** Roombots © 2010 IEEE, **(e)** ATRON[2], and **(f)** Pebbles © 2010 IEEE. Reprinted from [58–63] respectively.

are quasi-spherical modules that form face-centred cubic lattices which they can move over through rotation [71]. Modules in the M-TRAN system comprise two cubic halves linked by an additional degree of freedom, enabling them to talk over regular arrangements of other modules [72, 73]; a similar motion is also possible with Roombots modules, whose two halves are spherical instead [74] (Figure 2.3d). Other systems, such as ATRON [62] (Figure 2.3e) or UBot [75], only contain a single rotational degree of freedom so are only capable of more limited motions on their own. It should be noted that, while these systems are typically lattice-based, it is also possible for those with internal rotational joints to form structures resembling chain-type systems. However, they are not considered chain-type as individual modules cannot move outside of the lattice.

Finally, *fixed* lattice systems do not incorporate any internal actuation or degrees of freedom. Modules are instead actuated by some external force. Systems such as Miche [76] or Pebbles [63] (Figure 2.3f) rely on gravity to remove them from initial configurations made by a human. Others rely on stochastic excitation of a fluid, either water [77] or air [78], to self-assemble. Individual modules can influence the self-assembly process by choosing which connectors to activate. A wide range of structures can be formed, but the potential applications outside the laboratory are limited.

[2]Reprinted with permission from E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund, "Design of the ATRON lattice-based self-reconfigurable robot," *Autonomous Robots*, vol. 21, no. 2, pp. 165–183, 2006

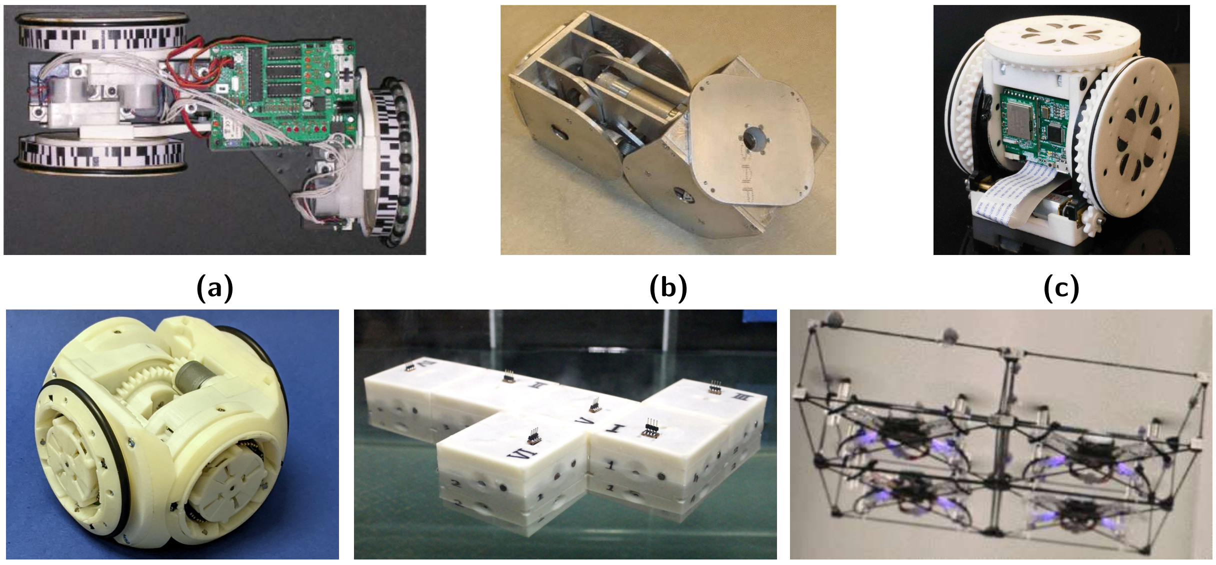


Figure 2.4. Examples of mobile self-assembling modular robotic systems: **(a)** M^3Express © 2012 IEEE, **(b)** iMobot © 2010 IEEE, **(c)** SMORES-EP[3], **(d)** HyMod, **(e)** MHP © 2016 IEEE, and **(f)** ModQuad © 2018 IEEE. Images excluding **(d)** reprinted from [81–85] respectively.

2.2.1.C Mobile

The third class of reconfigurable modular robotic system is the mobile class. These modules can move around their environment independently, and come together to form chain- or lattice-type arrangements. One of the first self-assembling modular robots designed, CEBOT, adopted a mobile topology [79]. These modules could drive around a flat surface using wheels, and dock with other modules to form different structures. Later systems such as M^3 [80] and M^3Express [81] (Figure 2.4a) also employ wheels to allow individual modules move around the environment independently. These systems comprise a pair of axially-aligned wheels and third orthogonal wheel, with connectors on the face of each wheel.

Many systems allow modules to work together as a collective after self-assembly. Some robots are able to demonstrate collective behaviour once assembled despite forming rigid structures without any internal actuation: for example, the PuzzleBots can cross gaps too large for an individual agent to drive over [86]. However, internal actuation such as the wheels of the aforementioned M^3 and M^3Express is often incorporated to allow the self-assembled structures to exhibit additional functionalities, such as to enable a hexapot configuration to walk. Other actuated degrees of freedom commonly included to achieve this is are central rotating joints. One of the first mobile self-assembling robotic systems to incorporate these was iMobot [82] (Figure 2.4b). Modules in this system resemble those of M-TRAN, but with additional rotating faceplates at their ends. These robots are able crawl with an inchworm gait using their central rotating joints, while the faceplates enable to them to rotate on the spot or can be used as wheels in a differential drive arrangement.

[3]Reprinted with permission from C. Liu, Q. Lin, H. Kim, and M. Yim, "SMORES-EP, a modular robot with parallel self-assembly," *Autonomous Robots*, vol. 47, no. 2, pp. 211–228, 2023

A popular contemporary design is for modules to be roughly cubic, and consist of two halves which rotate with respect to each other. Modules have connectors on four of the six faces, while the other two are empty to allow space for this rotation. Wheels are used both to allow modules to move around their environment and to actuate the self-assembled structures. A differential drive arrangement is common, as in SMORES [87], the second-generation SMORES-EP system [83] (Figure 2.4c), and HyMod [25] (Figure 2.4d). Other platforms use different arrangements of wheels to achieve a wider range of motions: for example, Omni-Pi-Tent uses omnidirectional wheels [88], and CoSMO uses Archimedes Screws [89].

The aforementioned systems are all land-based, but self-assembly is also possible in fluids. Robotic platforms designed to operate in these environments typically exhibit mobile architectures to take advantage of their low friction nature. For example, the modular hydraulic propulsion (MHP) system consists of square modules which float on the surface of water [84] (Figure 2.4e). They can route fluid through internal pumps to move individually, or connect with other modules to create an increased number of internal fluid pathways and move through complex trajectories [90]. Other systems form flying structures: modules in the Distributed Flight Array first self-assemble structures while driving on wheels before flying as a group [91], while the ModQuad can self-assemble into different structures when airborne [85] (Figure 2.4f).

2.2.1.D Freeform

In the previous three categories of self-assembling modular robot, connections between modules must be made at discrete points. Freeform systems do not have this limitation, and instead allow individual modules to dock with each other at any location. This increases the versatility of the system by allowing a wider range of configurations to be formed than the other topologies, and reduces the need for precise alignment between connectors. The Swarm-Bots project developed one of the earliest examples of this type of system [96] (Figure 2.5a). These robots could drive around a range of surfaces using a combination of tracks and wheels. When required, they could use one or two grippers to attach to adjacent agents at any point on a ring encircling them. Another system developed at around the same time was Slimebot [97]. These robots were roughly hexagonal and coated with Velcro to form connections with each other at any location. A more recent 2D freeform system is *Eciton robotica*. These are soft-bodied robots that have corkscrew grippers to attach themselves to Velcro surfaces, including other modules [93, 98] (Figure 2.5b). Cables running through the soft bodies allow them to deform and move with a flipping gait, but current designs only allow for controllable motion in 2D, so the modules operate in a vertical arena between two clear panes of acrylic.

Recent freeform modular robots can self-assemble 3D structures. One method of achieving this is to create modules out of ferromagnetic shells, which can join to other modules at any location through magnetism. The first robot to utilise this design was FreeBOT, which uses a carriage within a ferromagnetic sphere to drive around flat surfaces of any material, and vertical surfaces made of smooth ferromagnetic materials, including other modules [94] (Figure 2.5c). The concept has since been expanded to create other robotic platforms. These include FreeSN, a heterogeneous system comprising active strut modules that drive around passive spherical ferromagnetic node modules [99], and Snail-Bot, in which a wheeled carriage is placed on the outside of a ferromagnetic sphere to

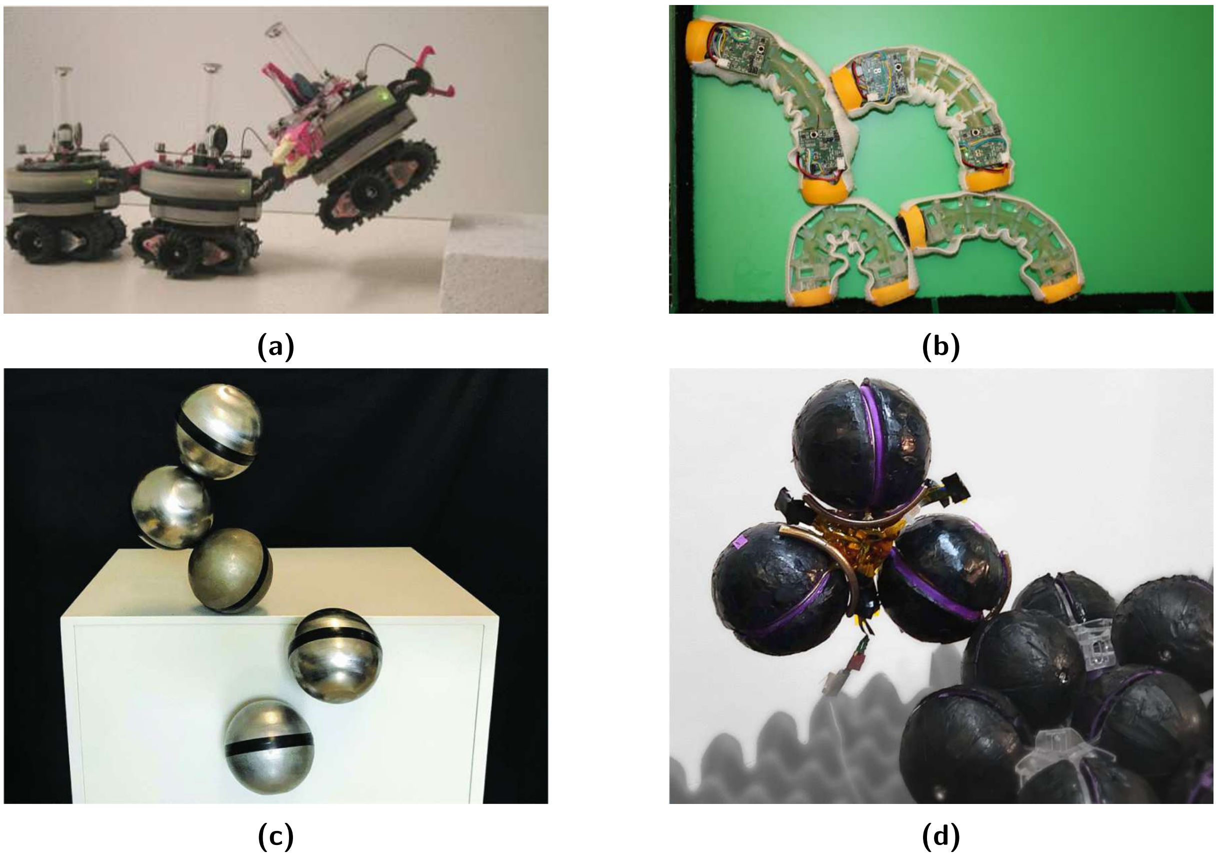

(a) **(b)** **(c)** **(d)**

Figure 2.5. Examples of freeform self-assembling modular robotic systems: **(a)** Swarm-bots © 2006 IEEE, **(b)** *Eciton robotica* © 2020 IEEE, **(c)** FreeBOT © 2020 IEEE, and **(d)** FireAnt3D © 2020 IEEE. Reprinted from [92–95] respectively.

self-assemble structures in a similar manner [100].

Another 3D freeform robot is FireAnt3D [95] (Figure 2.5d). Based on a previous 2D system [101], these robots comprise three spheres arranged in a triangle. The spheres are coated in a special polymer which melts when current is passed through it due to Joule heating, causing modules to become bonded to each other. Modules can walk over each other with a flipping gait, but can only move across surfaces coated in this special polymer.

2.2.2 Connection Methods

Self-assembling modular robots should be able to form secure connections with their peers to enable strong and functional structures to be built. Different methods of managing these connections are described below, categorised by the technology used to achieve the connection. The majority of designs either exploit interlocking *mechanical* components or *electromagnetic* interactions. A third section describes the small number of systems which rely on the *phase change* of materials as they change temperature to form connections.

Connectors are often classified based on how the design enables modules to interact. The three classes described by Parrott *et al.* [26] are:

- **Gendered:** Modules feature two distinct kinds of connector, typically one active and one passive. They can only form connections with the opposite kind.
- **Bi-gendered:** Active and passive components are combined into each connector, so that connections can be formed between all connectors.
- **Genderless:** All connectors feature active components. This allows either connector forming a connection to disconnect without actuation from the other side, something not possible in the other classes. This ability allows broken or otherwise non-functional modules to be removed from the self-assembled structure.

These terms are used to describe the connectors discussed below. In addition to the gender, other considerations must be made by designers. Factors such as the strength of the connections, the speed with which connections are made, the energy required during connection and while connections are active, and the tolerance of the connectors to misalignment are all important, and are discussed accordingly below.

2.2.2.A Mechanical Connectors

Some of the simplest mechanical connections revolve around the insertion of a peg into a hole. In systems such as CEBOT [79] and CONRO [47] (Figure 2.2c), latches are actuated when insertion is complete to lock the two parts together. These connections are strong, but the gendered design restricts the range of allowable connections. Furthermore, the latch means only one module is able to disengage the connection: if this module malfunctions, the connection cannot be broken. A similar design is employed by the CoBoLD connector [102] (Figure 2.6a) incorporated into the CoSMO system [89]. Each connector features both pegs and holes, as well as a rotating latch. This bi-gendered design allows them to attach to any other connector, but they can only disconnect if both latches are released. Latches such as these only draw power while being actuated, instead of requiring a constant power draw while connected. This design also includes force-sensitive resistors to measure the forces in the connector. Similar measurements are obtained by the connectors in the CHOBIE II system (Figure 2.3b), but strain gauges are used instead of force-sensitive for their increased reliability [67].

More recent systems have also used peg-and-hole connectors, but with novel features. The PuzzleBots platform incorporates a bi-gendered design where modules position pegs within holes, which then engage automatically due to gravity when one module drives over a gap [86] (Figure 2.6b). Mori also employs a bi-gendered design, where modules align male and female components on coupling axes on their edges. Latches extend to connect these parts that are designed to slip if torques in the connector exceed safe limits to prevent mechanical damage [103] (Figure 2.6c). The latches are extended using shape-memory alloy, whereas most modern mechanical connectors use electric motors to actuate.

Extendible hooks have become a popular design for mechanical connectors. This is the approach taken by M-TRAN III [73] and Roombots [106] (Figure 2.6d). M-TRAN III incorporates a gendered design, where half of the connectors on each module contain extendible hooks that engage with specially designed holes on opposing connectors. Roombots connectors are bi-gendered, with each connector incorporating both hooks and slots for them to engage with. Later designs also include include permanent magnets to aid with the alignment, but not to provide any significant additional strength [7]. The

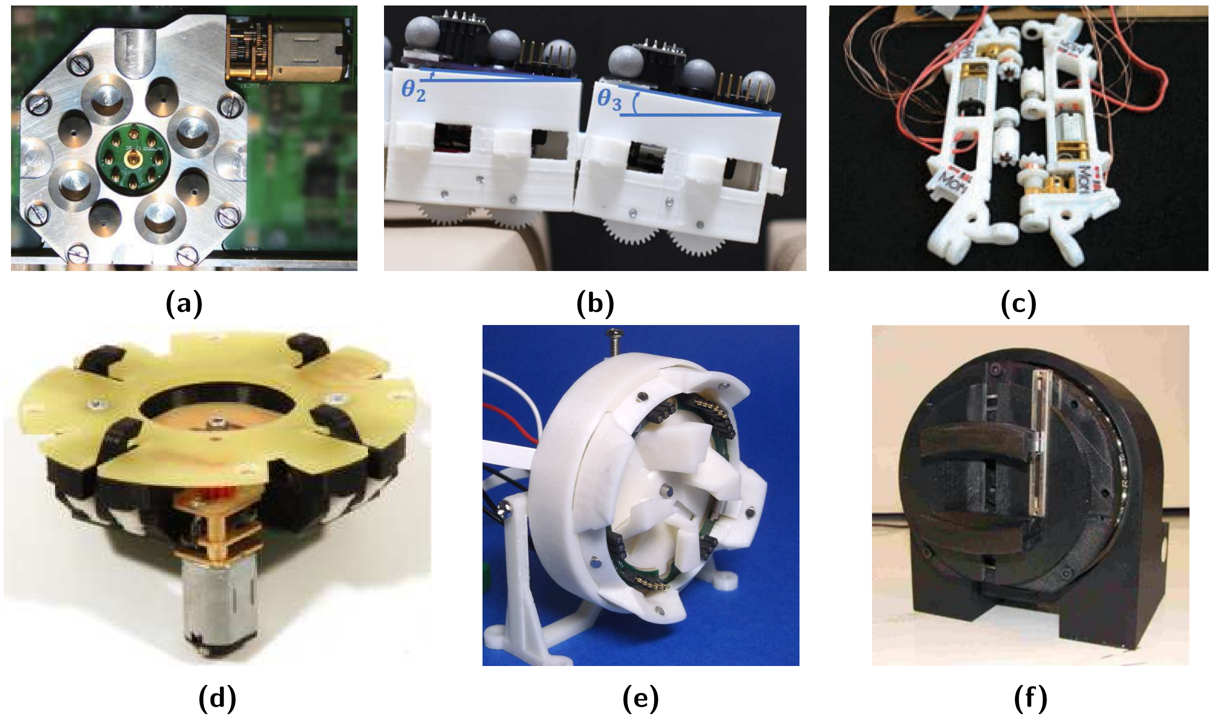


Figure 2.6. Examples of mechanical connectors for self-assembling modular robotic systems: **(a)** CoBoLD © 2011 IEEE, **(b)** PuzzleBots © 2021 IEEE, **(c)** Mori © 2019 IEEE, **(d)** Roombots[4], **(e)** HiGen © 2014 IEEE, and **(f)** GHEFT[5]. Reprinted from [26, 86, 102–105] respectively.

hooks of each of these platforms are designed to not be able to rotate unless driven by the actuating motors, so the motors can be turned off once connections are established to save energy. They also do not extend far from the surface of the module, meaning modules must slide against each other when aligning before connection. The hooks of ATRON modules extend further from the body, leaving gaps between adjacent connected modules to allow them to rotate within lattice arrangements without colliding with their neighbours [62].

Further mechanical connectors use hooks in a different manner to the above systems. Instead of engaging with female slots on opposing connectors, the docking hooks in the RoGenSiD connector [107] used in the ModRED platform [48] (Figure 2.2d) interface with identical docking hooks on opposing connectors. Hooks are rotated into contact during connection, meaning that reversing the rotation of either set releases the link, thus demonstrating a genderless design. However, the modules only fully disengage when the hooks are moved apart from each other: ModRED achieves this through a translational degree of freedom within each module, while the similar connector incorporated by Omni-

[5]Reprinted with permission from A. Spröwitz, R. Moeckel, M. Vespignani, S. Bonardi, and A. J. Ijspeert, "Roombots: A hardware perspective on 3D self-reconfiguration and locomotion with a homogeneous modular robot," *Robotics and Autonomous Systems*, Reconfigurable Modular Robotics, vol. 62, no. 7, pp. 1016–1033, 2014

[5]Reprinted with permission from W. Saab and P. Ben-Tzvi, "A Genderless Coupling Mechanism With Six-Degrees-of-Freedom Misalignment Capability for Modular Self-Reconfigurable Robots," *Journal of Mechanisms and Robotics*, vol. 8, no. 6, 2016

Pi-Tent requires modules to drive away on their wheels [88]. The HiGen connector [26] (Figure 2.6e) used in the HyMod platform [25] (Figure 2.4d) uses similar rotating docking hooks, but they are also extendible. This means that, similar to ATRON, HyMod modules can perform in place lattice rotations without collision. This connector is also significantly faster than RoGenSiD, taking only 0.2 s to actuate, compared to 12 s. RoGenSiD is slower as the docking hooks are driven through a worm gear, while in HiGen they are connected directly to the motor. This has the added effect of making RoGenSiD non-backdrivable, whereas the docking hooks on HiGen can rotate when a moment is applied to them: to ensure connections do not open prematurely, the controller constantly checks the rotation of the hooks and occasionally corrects the position through short motor pulses. This slightly increases the power draw of HiGen compared to RoGenSiD while connections are active.

Clamps can also be used to create genderless mechanical connections. The SuperBot platform [108] uses the SINGO connector [109], which has four orthogonal tracks on its connection face. Jaws move linearly along these tracks driven by a single motor, and clamp against jaws on the opposite tracks. A similar design is employed by the GHEFT connectors [105] (Figure 2.6f) used by the STORM system [110]. This design uses only two clamps which are actuated by a constant lead cam, and provides greater tolerance to misalignment of the connectors than SINGO. Both designs are not backdrivable, thus remain strongly connected without constantly drawing power from the actuating motor. Although they are genderless, they require modules to decide before connection which jaws will be on the inside of the connector and which will be on the outside, so they require additional coordination compared to other genderless designs, such as HiGen.

2.2.2.B Electromagnetic Connectors

The simplest method of using electromagnetic forces to connect self-assembling modular robots is to use *permanent magnets*. Connections are made quickly and require no energy to establish or maintain. However, the attractive force cannot be controlled, so detaching modules requires careful consideration. The attractive force of permanent magnets decreases rapidly as the distance from the magnet increases, so connectors employing permanent magnets typically disconnect by mechanically moving the opposing magnets apart from each other. In the M-TRAN and M-TRAN II systems, shape-memory alloy is used to bring four magnets per connection face towards and away from the edge of the module [72]. This creates a simple mechanism, but each actuation takes over a minute. All magnets in a given face are oriented with the same pole facing out, creating a gendered connection. In later systems, such as SMORES, each connection face incorporates a radially symmetric arrangement of polarities to create a bi-gendered connector [87] (Figure 2.7a). SMORES modules also have a retractable arm which can protrude from the centre of one connector at a time for two purposes. Firstly, it enables one face within a connection to remain stationary while the other is rotated to move the magnets away from each other and cause disconnection. The second purpose is to provide shear strength, as flat permanent magnets are typically able to withstand high axial loads but are weak to shear forces. Other designs that rotate magnets away from each other to control connections include PPT [111] and Evo-Bots [78]. The freeform connectors based on ferromagnetic shells employed in FreeBOT [94] (Figure 2.5c) and FreeSN [99] rely on an internal carriage to raise and lower permanent magnets to control the connection, while SnailBot achieves a

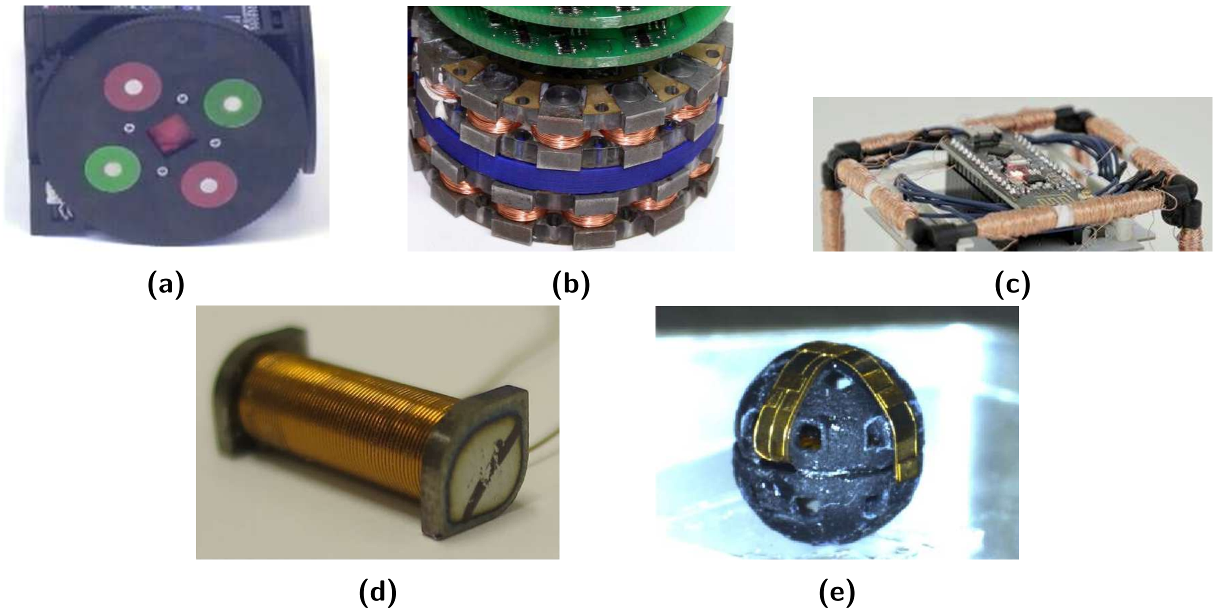


Figure 2.7. Examples of electromagnetic connectors for self-assembling modular robotic systems: **(a)** SMORES © 2012 IEEE, **(b)** Catoms © 2007 IEEE, **(c)** ElectroVoxels © 2022 IEEE, **(d)** Kubits © 2020 IEEE, and **(e)** 3D Catoms © 2022 IEEE. Reprinted from [69, 70, 87, 112, 113] respectively.

similar effect through its external chassis [100]. In these freeform designs, the connection is controlled by a single module, as the ferromagnetic sphere is unable to break connections made to it. The final modules incorporating permanent magnets that is mentioned here are 3D M-Blocks [60] (Figure 2.3c): these break the magnetic connections simply by applying a large torque to overcome the attractive forces.

Applying an electric current to a coil of wire creates a magnetic field that can be controlled. Such devices are called *electromagnets*, and have been used in self-assembling modular robots as they allow small, robust, strong, and versatile connectors. Some systems, such as the original Catoms [112] (Figure 2.7b) and ElectroVoxels [70] (Figure 2.7c), create connections using only this phenomena. Others, such as Fracta [55], Molecubes [114], and EM-Cubes [68], use a combination of electromagnets and permanent magnets. The latter arrangement reduces energy usage, as fewer electromagnets must be powered in each active connection.

An important downside of electromagnets is that they constantly draw energy while activated. This is not the case for *electropermanent magnets* (EPMs). These are effectively permanent magnets with a coil wrapped around them, through which a current flows. The magnetic field of the permanent magnets can be turned on and off by pulsing current through the coils, and remains in this state when the current is removed. This therefore reduces the energy consumed by the connection. The first modules to use this technology were the Pebbles in 2010 [63] (Figure 2.3f), and it has since been used by Lily [77], DONUTs [115], Kubits [69] (Figure 2.7d), and the EP-Face connector [116] of SMORES-EP [83] (Figure 2.4c). These connectors are all genderless, but require modules to coordinate to decide on the polarity of the EPMs before connections can be formed.

In addition to magnetic forces, the attraction between charged particles is also an electromagnetic phenomenon that can be used to form connections between self-assembling modular robots. This has been shown at large scales by Karagozler *et al.* [117], who demonstrated an electrostatic connector capable of lifting a module weighing 3.5 kg. This technology shows its greatest potential at small scales as it can be effectively miniaturised, as demonstrated by the genderless connectors on both 2D Catoms [118] and 3D Catoms [113] (Figure 2.7e).

2.2.2.C Phase Change Connectors

Instead of employing interlocking mechanical components or utilising electromagnetic forces, a third option is to apply heat to certain areas of the connectors, causing them to melt and fuse together. This was first demonstrated by the Soldercubes [119], a system where exposed *printed circuit boards* (PCBs) are placed on the connection faces. Pads on these PCBs are coated with solder, which melt when a current is passed through them. As the solder cools, connectors placed in contact with each other are strongly bonded together. This system inspired the design of the freeform FireAnt [101] and FireAnt3D [95] (Figure 2.5d) platforms, described previously. The main advantages of this connection method are the high strength that can be achieved and the inherent genderless properties of this design, but connections are typically slow to form as the material must cool sufficiently before they reach their full strength.

2.2.3 Actuation Methods

Self-assembling modular robotic systems should include some amount of actuation. This is used both for modules to move around the environment, and to actuate the self-assembled structures to enable them to complete different tasks. The choice of actuation methods included within a given platform is related to its topology: for example, chain-type and mobile systems require a method of achieving locomotion independent of other modules, while lattice-type systems only need to consider actuation relative to their neighbours. For this reason, methods of actuation for the platforms mentioned in Section 2.2.1 are only briefly covered again here.

There are several common methods of actuating self-assembling modular robotic systems, discussed below. Some platforms rely on *external actuation* to simplify the design of each module. A common method of actuating mobile systems is to use *wheels*, while platforms of all topologies employ *internal joints* to actuate the self-assembled structures and for the locomotion of modules around the environment. *Electromagnetic forces* may also be used to provide actuation, which is particularly attractive for systems that use this style of connector, as the same hardware can be used for both purposes. Finally, a small number of *other methods* are mentioned.

2.2.3.A External Actuation

The simplest designs for self-assembling modular robotic platforms rely on external actuation to move individual agents. The earliest examples of engineered self-assembly utilised external actuation to induce controlled pattern formation in passive building blocks. Such systems, including those developed by Penrose and Penrose [120], typically consist of an arena in which the agents operate which can be actuated by an external force.

PPT [111] and Evo-Bots [78] operate on air tables and are propelled by air currents created by the random motion of fans on the edge of the table. The Lily robots are placed on the surface of a liquid, which is agitated with a pump to induce them to move around the arena [77]. In another example by Jilek *et al.* [121] the arena is attached to a robotic arm programmed to move through an inclined elliptical path, which causes the robots to slide around the arena floor. Other systems use gravity to provide the actuation in a more simple way: systems such as Miche [76] and Pebbles [63] (Figure 2.3f) are first assembled by hand into a connected structure, then agents choose to release their connections and fall away from the structure, leaving behind a desired shape.

Externally actuated systems such as those mentioned above are relatively inexpensive, robust, and allow for large numbers of robots to be deployed to investigate the scalability of control algorithms. However, their potential real-world applications are rather limited due to the specialist arenas they are required to operate in.

2.2.3.B Wheels

A popular method for modules to move around their environment independently is to use wheels. The most common configuration is to use a pair of wheels arranged as a differential drive, where each wheel can be controlled independently to allow modules to be highly manoeuvrable. This configuration is used by CEBOT [79], M^3 [80], M^{3}Express [81] (Figure 2.4a), iMobot [82] (Figure 2.4b), SMORES [87], SMORES-EP [83] (Figure 2.4c), HyMod [25] (Figure 2.4d), and PuzzleBots [86] (Figure 2.6b) as described earlier.

Other systems use different arrangements of wheel-like mechanisms to locomote. For example, Swarm-bots use a combination of tracks and wheels to perform efficient on-the-spot rotation and to allow for simpler navigation than regular tracks or wheels [96] (Figure 2.5a). STORM also uses tracks and wheels, but in a different manner: normally, motion is achieved through the use of tracks, but a set of wheels oriented orthogonal to the tracks can be raised or lowered to allow agents to also move perpendicular to the tracks, a feature that is particularly useful while docking with their peers [110]. Other methods of moving in different directions without rotating the robot body include the omnidirectional wheels of Omni-Pi-Tent [88], and the Archimedes Screws of CoSMO [89]. FreeBOT can also drive in any direction without any apparent force applied to the external ferromagnetic shell as the internal carriage can rotate unseen on wheels to move the module around the environment [94]. The related FreeSN [99] and SnailBot [100] systems also use wheels to drive on ferromagnetic surfaces.

In addition to providing a method of locomotion for individual robots, wheels also act as additional degrees of freedom for self-assembled structures. Incorporating the connection mechanism into the wheels is a popular design choice made by M^3 [80], M^{3}Express [81] (Figure 2.4a), iMobot [82] (Figure 2.4b), SMORES [87], SMORES-EP [83] (Figure 2.4c), HyMod [25] (Figure 2.4d), and STORM [110].

2.2.3.C Internal Joints

Many robotic platforms include internal joints, both for individual locomotion and to enhance the functionalities of self-assembled structures. Some systems, including Crystalline [64] (Figure 2.3a) and Telecubes [65], incorporate prismatic joints. However, these joints are typically rotational as these offer more flexibility and are easier to incorporate into small robot bodies. Occasionally, the robot morphology allows agents to use these joints to independently crawl along flat surfaces, as has been shown for ModRED [48] (Figure 2.2d), iMobot [82] (Figure 2.4b), and SuperBot [108]. *Eciton robotica* agents can deform their soft bodies to locomote by flipping [93, 98] (Figure 2.5b), while FireAnt3D can also move in a similar manner [95] (Figure 2.5d); both these freeform connectors can only attach to specific surfaces though, so these robots cannot move in arbitrary environments.

Individual locomotion through internal joints is achieved most successfully in lattice-type systems, as the lattice simplifies the alignment of connectors, and existing modules provide a solid base for locomoting agents to rotate around. Modules that span across two lattice cells can achieve this on their own, such as iMobot [82] (Figure 2.4b), SuperBot [108], or Roombots [74] (Figure 2.3d). Other systems consist of modules that each occupy a single cell in the lattice so must form a *metamodule* of at least two agents to move across the lattice in this manner: such systems include ATRON [62], UBot [75], SMORES [87], SMORES-EP [83] (Figure 2.4c), and HyMod [25] (Figure 2.4d).

When several modules are connected together, internal joints allow for the structure to be carefully actuated as a whole. Several platforms have demonstrated collective motions in this manner with a variety of morphologies and gaits, such as crawling [122], snake-like motions [49], rolling loops [45, 123], and walkers [72, 123, 124]. They can also use these abilities to create structures such as robotic manipulators [7, 25, 125]. Furthermore, the rotational degrees of freedom can be used as wheels if designed in a suitable manner, as shown in Roombots [7] and ATRON [125].

2.2.3.D Electromagnetic Forces

Systems that employ magnetic connectors can reuse the same hardware to provide actuation. This reduces the mechanical complexity of each module, making them less expensive and easier to manufacture. The basic principle of such systems is to set certain magnets to attract and others to repel, creating a force that induces a rotation relative to fixed point. Fracta was one of the first systems to use this effect to move agents to adjacent lattice positions [55], and it has since been demonstrated with other connectors that utilise electromagnets, such as Catoms [112] (Figure 2.7b) and ElectroVoxels [70] (Figure 2.7c). The same principle can also be used with EPMs, as demonstrated by DONUTs [115] and Kubits [69] (Figure 2.7d). The EM-Cubes [68] use electromagnets to create translational motions in contrast to the rotational movement exhibited by other systems.

This approach is also applicable to electrostatic connectors. 2D Catoms demonstrated how modules can rotate in relation to one another by controlling electrostatic charges [118], and the same approach is utilised by 3D Catoms [113] (Figure 2.7e).

2.2.3.E Other Methods

The 3D M-Blocks [60] (Figure 2.3c) are a cubic lattice-based platform with a unique method of locomotion. These modules pivot around their edges by first rotating an internal flywheel up to a high-speed then applying a sudden brake. The inertia causes the module to rotate in the direction the flywheel was spinning, and permanent magnets on the edges ensure the rotation occurs about that axis. The mechanism was first demonstrated in 2D [126], before it was modified to allow the flywheel to be reoriented and thus enable modules to move in 3D.

Robotic systems that operate in fluids face unique challenges due to the low friction nature of their environment. Aerial systems typically use rotors to generate lift, controlling their movement by varying the speed and pitch of their blades [85, 91]. The MHP platform operates on the surface of water and uses internal pumps to route fluid within robots to control the motion of either individual agents or a self-assembled collective [90].

2.2.4 Communication Methods

The final choice considered by engineers when designing self-assembling modular robots is how the agents should communicate, either with their peers, a human operator, or both. The different approaches that are taken can be split into two categories: those that use *electrical contacts*, and those that employ *wireless methods*.

2.2.4.A Electrical Contacts

Using electrical contacts is attractive as they allow for direct communication between neighbours, good signal quality, and high bandwidth. This approach was popular with early systems, partly because consumer wireless technology was still in its infancy when these systems were developed around the turn of the millennium. For example, each connection face on M-TRAN featured five electrical contacts: four were used for power and ground connections, while the fifth was used to transmit serial data between modules and to communicate with a controlling computer [72]. CEBOT realised parallel data transmission between docked modules through a 14 pin connector [79].

The above systems use custom communication protocols. However, electronic engineers have developed a number of standards to enable wired communication between devices. By using these standards, researchers can benefit from robust communication protocols and focus on other factors in the design of robotic platforms. For example, PolyBot used CAN in 2001 [127], a protocol that uses two conductors to enable communication between microcontrollers. This is also used by later systems, such as HyMod [25]. Another popular communication protocol used in networking is Ethernet. It enables high-bandwidth communications, but requires more wires than CAN. Nevertheless, it has been used in a small number of self-assembling modular robotic platforms such as CoSMO [89] and that of White *et al.* [128].

Forming reliable electrical connections between modules requires each module to be carefully aligned and strongly held in place. This imposes restrictions on the physical design of the platform, but can be beneficial for other purposes. An important benefit of forming electrical connections between modules is that power can be shared across these interfaces, in addition to data. This increases the operating time of the robots, as modules that are low on power can draw from their neighbours to avoid shutting down. This has been demonstrated for several platforms, including SuperBot [78] and Evo-Bots [129]. Reducing the number of electrical contacts is beneficial as it means they occupy less space on the connection faces. Power can be transmitted without any additional contacts through the use of standards such as *power over Ethernet* [128], or through careful reuse of pins, as shown by a method developed by Holdcroft *et al.* which uses just three wires to transmit power and high-bandwidth data [130].

Designing methods of forming electrical contacts is particularly challenging in freeform platforms as there are no specific defined contact points. However, FireAnt allows for wired communications through the contacts used to transfer current when heating the polymer coating [101]. Power sharing can be achieved in systems with ferromagnetic shells by using the shell as an electrical contact [131].

2.2.4.B Wireless Methods

Many platforms use wireless communication, as it places fewer restrictions on the design of modules than imposed by wired methods. The earliest self-assembling modular robots that could communicate wirelessly did so through infrared waves, as such systems are easy to use and widely available. Systems such as CONRO [47], ATRON [62], and SuperBot [108] all utilised infrared methods to communicate. The infrared LEDs used to transmit data can also act as beacons to aid modules during alignment. This technology has been less popular in recent years, but is still used by 3D M-Blocks [60] and Omni-Pi-Tent [88].

The downsides of infrared communication include its limited range and bandwidth, as well as its requirement for the transmitter and receiver to be in direct line of sight of each other. Developments in consumer wireless communication standards in recent decades have been embraced by designers of self-assembling modular robotic systems, offering several advantages over infrared methods. Swarm-bots were among the first to utilise Wi-Fi [96], benefiting from its high bandwidth, relatively long range, ease of deployment in the laboratory, and strong support from manufacturers and software developers. These agents also featured multicoloured LEDs and a camera to communicate visually. Wi-Fi has become very popular in modular robotic platforms recently, and has been used by systems including PuzzleBots [86], Omni-Pi-Tent [88], STORM [110], and SMORES-EP [83].

Other wireless standards have also been used in recent platforms. Bluetooth is a well-known standard used in consumer electronics, which has also been deployed in robotic platforms such as M^3 [80], M^{3}Express [81] and HyMod [25]. Another similar standard is XBee, as used in ModRED [48].

Wired and wireless approaches are each suitable in certain scenarios. For this reason, it is common for robotic platforms to use a combination of both methods: modules communicate with those they are connected to through wires, and with others wirelessly. Holdcroft *et al.* recently demonstrated a particularly capable hybrid approach, where bandwidth and reliability are improved by using a combination of wired and wireless protocols based on the type of data being sent [132].

The wireless communication methods mentioned above allow modules to communicate with each other from a distance. Wireless methods can also be used at short ranges to allow communication between adjacent modules without the precise alignment of electrical contacts. For example, the *Eciton robotica* agents communicate through vibration motors, which are used to send messages received by accelerometers [93]. Electromagnetic induction can also be used to allow modules to communicate wirelessly with their neighbours. This is particularly common in systems that use electromagnets or EPMs for connection or actuation, as the same hardware can be used to communicate: applying a current to the coil of one magnet will induce a similar current in the opposing coil. This method has been demonstrated by systems including Pebbles [63], Lily [77], DONUTs

[115], and SMORES-EP [116]. Messages can also be sent across electorstatic connectors in a similar manner, as demonstrated by Karagozler *et al.* [117] and used on both 2D [118] and 3D Catoms [113].

2.3 Control of Self-Assembling Robots

At the same time as the hardware required to perform robotic self-assembly is being developed, the design of algorithms to enable this functionality is also an active research area [133]. Approaches are often divided depending on whether they use *centralised* control, where a single leader instructs other agents what operations to perform, or *decentralised* control, where agents make their own decisions. The majority of recent developments have taken a decentralised approach, due its resilient and scalable nature [134]. The literature is therefore divided here by the purpose of the algorithms, inspired by Stoy *et al.* [2]. In *shape-driven* self-assembly, robots are tasked to create a shape specified by a human user, whereas in *task-driven* self-assembly, the robots are given a high-level task to complete, from which the structure emerges. A further category of *auxiliary algorithms* includes low-level procedures that are necessary to achieve the behaviours taken as a prerequisite by more high-level algorithms.

The self-assembled structures are often required to actuate themselves, for example to enable them to walk [50, 135]. This thesis considers the self-assembly process itself, so the algorithms to enable such coordinated actuation once self-assembly is complete are not considered here.

2.3.1 Shape-Driven Self-Assembly

Some of the earliest developments into self-assembly algorithms took a shape-driven approach, often tailored to the geometry of the hardware they were implemented on. For example, algorithms developed for the Fracta system allowed reconfiguration into desired shapes in a distributed manner [136], while later research investigated 3D reconfiguration using the Proteo platform [137]. The former work used a set of pre-compiled local rules followed by each agent, while the latter used a "goal-ordering" approach, where modules attempt to reach goal locations assigned in different manners. Other algorithms developed to reflect the morphology and capabilities of specific robotic platforms include work with ATRON to find optimal module paths during reconfiguration of structures with relatively few modules [138], and methods of self-assembling Roombots modules into structures resembling furniture such as chairs [139].

In addition to algorithms developed with a specific platform in mind, researchers are often interested in developing algorithms for a generic class of hardware. In these cases, low-level abstractions are commonly used. One such common abstraction is the *sliding cube* model, popularised by Fitch *et al.* who used it to demonstrate an algorithm that could create arbitrary shapes by first disassembling an existing self-assembled structure into a line before the goal structure was built [140]. In this model, modules are cubes that can slide over their peers in either adjacent or convex translations as shown in Figures 2.8a & 2.8b. Recent self-reconfiguration algorithms that use this model of object movement include methods based on virtual forces [141] or Lindenmayer systems [142],

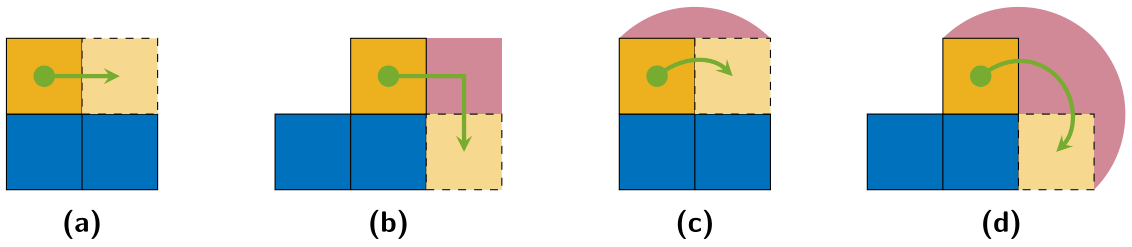


Figure 2.8. The sliding **(a − b)** and pivoting **(c − d)** cube models, shown in 2D. Each yellow agent follows the green arrows to move to their next location, shown paler. In doing so, they sweep out the red areas, which therefore must be empty to avoid collisions. **(a)** and **(c)** show adjacent translations, while **(b)** and **(d)** show convex translations.

or a layered reconfiguration process that starts from outermost layer [143]. These works are all implemented in 2D: reconfiguration in 3D remains a challenge.

The sliding cube model is difficult to realise with physical systems. Adjacent translations (Figure 2.8a) can sometimes be achieved by single modules, as in translational lattice platforms such as Crystalline [64] (Figure 2.3a) and CHOBIE II [67] (Figure 2.3b), or by a metamodule of agents from a pivoting lattice system. However, convex translations (Figure 2.8b) are more complex, often requiring assistance from additional modules within the structure. One approach to tackle this is to use a 2 x 2 x 2 metamodule comprised of modules that can independently perform the adjacent translation to achieve the more challenging convex translation together. First shown in a centralised manner [144], work has now progressed into a distributed algorithm more suitable for real robotic hardware [145].

A more realistic hardware abstraction is the *pivoting cube* model, which encapsulates how cubic robotic systems such as 3D M-Blocks [60] (Figure 2.3c), Kubits [69], and ElectroVoxels [70] move. It has different restrictions on module motion due to the area swept out during rotation (Figures 2.8c & 2.8d). An interesting theoretical development has shown that the reconfiguration problem based on this model is NP-complete [146], something previously shown for chain-type modular robots [147] but still not proved for the sliding cube model. Researchers have simplified this problem by only allowing a single module to move at once, and to convert the structure into an intermediary line configuration before the desired shape is built [148]; this approach has also been extended to allow the construction of structures with non-convex holes [149]. These algorithms rely on global control to avoid collisions, whereas other work has shown simple behaviours for pivoting cube robots in a decentralised fashion [150].

In addition to the precise manner in which modules move, further physical considerations behind self-assembly can be accounted for to improve the real-world applicability of these algorithms. For example, the most time-consuming step of reconfiguration with SMORES-EP (Figure 2.4c) is the docking and undocking, so algorithms that reduce the number of these actions required have been developed [151]. Self-assembly also sometimes occurs in cluttered environments, prompting researchers to incorporate obstacle avoidance strategies into their self-assembly algorithms [152].

The Programmable Matter consortium have proposed a robust method of achieving 3D self-assembly using 3D Catoms [153] (Figure 2.7e). Agents are added to the environment from a supply called the *sandbox*, and first build a minimalist approximate structure called a *scaffold* [154]. This facilitates the motion of modules within the structure, and minimises the number required to represent a given shape. The scaffold is then coated with a thin layer of agents to produce the final configuration [155]. This approach only describes the construction of an initial design from the sandbox, so any changes would require dismantling and completely rebuilding the structure. A more efficient approach is described in [156]: similar hollow structures are created using metamodules of ten 3D Catoms which can move through unoccupied locations within the bounding box of other metamodules and wait in specified locations so they can be used to reconfigure the structure at a later time. Large, low density, metamodules such as these are a promising method of producing complex motions, such as tunnelling through occupied locations, using modules with relatively limited motion capabilities [157].

Stochastic self-assembly is another promising research area, in which robots do not control their motion but instead choose what connections they make. This approach is massively parallel, creating scalable and robust systems [158]. The Lily platform has supported a number of works, including self-assembly of defined structures by generating rules for which connections should be retained when modules are stochastically brought into contact with each other [159]. The process has also been described using Markov models to gain insights into the behaviour of such systems [160]. Jilek *et al.* have also demonstrated tile-based self-assembly in a similar manner [121].

Instead of creating shapes using self-assembly, a contrasting approach is to use self-*dis*assembly. Robots start from a highly connected structure, then selectively remove themselves to leave a desired shape. This was first proposed for the Miche platform [76] and more recently demonstrated with Kilobot robots [161]. Modules in the initial structure are guaranteed to be able to communicate with each other, so the reconfiguration process can be decided upon in a centralised manner before movement commences. Early work relied on external forces to remove the modules [63, 76], whereas recent developments have shown how modules can remove themselves at a specified sink location to increase the versatility of this approach [162].

2.3.2 Task-Driven Self-Assembly

In task-driven self-assembly, a structure emerges as a result of modules attempting to complete a given task. This leads to self-assembly as modules automatically respond to the situation, instead of requiring separate pre-programmed shapes to be built to complete different tasks. Early work in this area was performed by Bojinov *et al.* [163]. They developed a framework whereby local rules are used with a finite-state machine on each agent to induce self-assembly of structures that fulfil a specified purpose. Four examples are presented to demonstrate the flexibility of this approach: construction of a chain, construction of a branching structure, grasping an object of unknown size, and reconfiguration of a table in response to a changing load. This last example uses measurements of the force within the connections between modules to guide the self-assembly process: the work presented in this thesis uses similar force-aware methods. For this reason, Section 2.4 is dedicated entirely to discussing methods of force-aware construction in the literature. The remainder of this section discusses other methods of guiding task-driven self-assembly.

The ATRON platform (Figure 2.3e) has been used to demonstrate task-driven self-assembly. Artificial neural nets were developed to build bridges and towers, as well as to repair broken structures [164]. This approach requires users to manually place *attraction points* in an arena containing simulated ATRON modules. These modules move towards the points in a manner controlled by the artificial neural nets. The final structures are not predefined by a human, instead arising as a result of the artificial neural nets and the positioning of the attraction points.

A common use of task-driven self-assembly is to allow a group of agents to move as a collective across unknown terrain, as demonstrated by the Million Module March algorithm [165]. Agents move to a target location by following local rules and deforming to the terrain, without requiring a user to specify what obstacles will be encountered. A similar algorithm was developed more recently by Zhu *et al.* and implemented with real-life robots [166]. The approach has also been applied to simulated freeform spherical robots based on FreeBOT [167], including making checks to ensure structures do not topple over [168].

O'Grady *et al.* demonstrated how a group of Swarm-bots can employ self-assembly to cross terrain they could not do so individually [169]. In these experiments, when an agent reaches a hill too steep to climb, it recruits nearby modules for assistance by changing the colour of its ring of LEDs. This change is seen by nearby agents, and they self-assemble into a group that can climb the hill together. The shape is not defined by a user, instead forming naturally as the robots try to aggregate; the freeform connector incorporated by the Swarm-bots helps with this amorphous shape formation. The same approach was also demonstrated to help the team cross a small gap in the terrain, and to rescue damaged agents.

Another recent demonstration of task-driven self-assembly considers the formation of bridges. Inspired by work investigating how ants self-assemble into bridges to enable them to shorten the path to a goal location [39] (Figure 2.1b), Malley *et al.* developed a task-driven algorithm to allow simulated robots to display similar behaviours [93]. Agents move along a V-shaped path to a goal, and can step on each other if necessary. While an agent is stepped on, it believes it is providing a useful shortcut and so remains stationary. When the agent is uncovered, it waits a short period of time before moving again in case another agent would also benefit from stepping on it. By varying the rate at which agents are added to the simulation and the angle of the V-shaped path, stable bridges of different lengths emerge, which act as shortcuts to allow other agents to reach the end of the track faster. The approach has also been extended to build structures to locations that can not be reached by a single agent [170].

2.3.3 Auxiliary Algorithms

In addition to algorithms that explicitly induce self-assembly, additional low-level algorithms may be required. For example, many approaches assume each module has a unique identity to coordinate with its peers, such as in the work of the Programmable Matter consortium [153]. Inter-module communications often contain the address of the intended recipient, which should therefore be compact to reduce the message length. A method for assigning these identities on static structures is presented in [171], and then extended to allow for dynamic structures by incorporating unassigned identities that can be used at a later time [172].

Many algorithms require agents to know their initial structure so they can decide how to move to a target structure. A method of discovering this has been demonstrated with the CKBot [53] and ModRED platforms [173]. The latter work is particularly noteworthy as it combines different methods of communication, incorporating infrared for local communication between neighbours and XBee to rapidly provide wireless updates throughout the system. Investigations have also been performed with SMORES-EP robots, where configurations are matched to an existing library to speed up reconfiguration [174]. In freeform modular robots, configuration recognition can be even more challenging, as connections are able to be made at any location. Nevertheless, an approach to solving this for FreeBOT has been developed [175], which models the magnetic field produced by the connectors using graph convolutional neural networks to determine the locations of the connections to each agent.

Other auxiliary algorithms consider problems that may occur in the real-world but not in simulation. For example, when robots break down they may require removal or replacement [88]. Alternatively, additional modules could be incorporated nearby to maintain functionality [176]. Another consideration is whether structures will collapse under their own weight, as in the problem of self-assembling bridges considered in this thesis; a method of rapidly predicting forces within self-asembled structures using distributed computation has recently been demonstrated [177].

2.4 Force-Aware Construction

This thesis considers how bridges can be self-assembled in a force-aware manner such that they do not collapse under their own weight. As mentioned in Section 2.3.2, this is an example of task-driven self-assembly. Inou *et al.* were some of the first researchers to investigate the force-aware self-assembly of bridges in 2000 [22]. They considered how a group of simulated agents operating in a 2D grid could self-assemble a cantilever to support a load as far from a vertical fixed support as possible. Their algorithm compares the calculated stress within each agent to three values:

- **Allowable:** the load moves away from the fixed support, and new agents are added at the tip of the cantilever as it does so, until the stress in an agent in the cantilever exceeds this limit.
- **Call:** if the stress in an agent exceeds this value, it is reinforced by adding another agent below it.
- **Out:** when the stress in an agent drops below this threshold, it leaves the structure.

The algorithm was tested for different values of the thresholds described above, different numbers of agents, and different loading configurations. It was found that the algorithm enabled the self-assembly of cantilevers in a robust manner that adapted to the current scenario. The stresses are calculated only once agents have been added to the structure, so the loads applied as agents move around the structure are not considered. Furthermore, only one agent moves around the structure at a time, making the self-assembly slower than if multiple agents could explore the structure at once.

Further work considered the communication capabilities required for agents to follow this self-assembly algorithm [23]. Whenever an agent is added to or removed from the structure, messages are passed between all agents within it to determine the next course of action. Each agent must pass messages to all of their neighbours to build a hierarchical tree describing the shortest route between the support and either the location where an agent will be placed or the agent that is being removed. This results in $\mathcal{O}(N^2)$ messages being passed to build a structure containing N agents, potentially limiting the scalability of the approach to large structures, and increasing the probability of communication failures occurring and affecting the operation of the algorithm.

Inou *et al.* also considered how these bridges could be deconstructed when the load has crossed to the other side of the gap. The same algorithm is capable of inducing deconstruction by relaxing certain constraints. However, several deadlocks occur: one in which deconstruction stalls when presented with certain configurations is resolved by adding additional position-based heuristics, while another where agents repeatedly add and remove from the same locations remains unresolved. Force-aware methods to deconstruct self-assembled bridges therefore remains an open challenge.

A promising recent development in force-aware self-assembly is the ReactiveBuild algorithm [24]. This algorithm enables the self-assembly of towers, chains, cantilevers, and bridges using local force measurements to guide the construction process. Agents add themselves to the structure one at a time, and move around it until they receive a message from a neighbouring agent that it is in need of reinforcement. The agent then joins the structure here, whereupon another enters the simulation and begins exploring. This algorithm was validated with simulated FireAnt3D modules, demonstrating how freeform platforms can benefit from a force-aware approach to self-assembly. Since agents are added at the first location they encounter that requires reinforcement, it is possible that they will ignore very high forces far from the root in favour of providing reinforcement to closer locations where forces are only moderately high. This effect limits the maximum size of structures that can be constructed.

A force-aware self-assembly algorithm has been developed for the translational lattice-type CHOBIE II platform [67] (Figure 2.3b). Agents include strain gauges to enable them to account for force measurements in the structure. Initial algorithmic developments for this platform enabled a group of CHOBIE II agents to move along a flat surface as a group, incorporating an important motion constraint of this system: when a module moves along a row or column, all the modules in that row or column must move with it [178]. This work also proposed how the strain gauges could be used to build cantilevers that do not break under gravity, but the idea was not explored in detail. Further developments produced distributed algorithms to enable modules to self-assemble into a given shape [179], and this framework was then extended to include considerations of the forces in the structure [18, 59]. The reconfiguration algorithm chooses leader modules based on how *undesirable* the current configuration is, a metric that is calculated from the positions of modules in its row and column compared to the desired configuration, as well as the current stress distribution in the structure. Chosen leader modules are permitted to move a row or column in the manner they believe would benefit the self-assembly process. This is a promising approach to achieving self-assembly without structural failure, but requires a high degree of coordination between the agents to move together. Furthermore, while the algorithms are developed with CHOBIE II in mind, the force-aware approaches are only demonstrated in simulation.

Researchers considering force-aware robotic construction in scenarios beyond self-assembly have also inspired the work in this thesis. Genetic algorithms were used by Funes and Pollack to produce cantilevers constructed from discrete building blocks that would not collapse under their own weight [15]. These structures were designed by a machine and built by hand: later work investigated how stable structures could be designed using a genetic algorithm, then built from wooden blocks and hot-melt adhesive by a robotic manipulator [16]. A related approach considers how the assembly sequence of a given structure by a robotic manipulator can be automatically derived by considering the structural stability at each step [14]. Multi-robot systems have also been used to demonstrate force-aware construction. The work of Melenbrink *et al.* has shown how robots can assemble truss-based cantilevers that do not collapse under their own weight [17, 180, 181]. A task-driven approach to the construction is taken, where agents react to the forces at the truss nodes to influence where subsequent assembly occurs. The process was validated in simulation, and proof-of-concept hardware was demonstrated, but extensive real-world trials were not performed.

2.5 Summary

In this chapter, an overview of the field of self-assembling modular robotics has been presented. A wide range of physical designs of such systems were first shown. As each has their own advantages and disadvantages, no single platform has emerged as the most capable. The traditional approach of designing modules with discrete connection points has been challenged in recent years by the development of freeform connectors, which offer wider-ranging configurations but with their own unique challenges relating to the increased configuration space they offer.

On the algorithmic front, a wide range of approaches are taken towards the self-assembly of specified shapes, or self-assembling structures to complete prescribed tasks; the latter approach may contribute to an increased utility of self-assembling multi-robot systems. Of particular note to this thesis are force-aware self-assembly methods. Such approaches have demonstrated how structures can be self-assembled that do not collapse under their own self-weight, without a human specifically designing their shape. Construction of cantilevers across a gap is the most common scenario examined, but actions to modify the structure when it becomes supported by the other side of the gap have not been considered. The reliable deconstruction of such structures remains a further open challenge. The approaches also only consider either the reconfiguration of a predefined number of agents, or situations in which only one agent at a time adds itself to the structure: algorithms where multiple agents can explore the structure in parallel with each other have not been developed. These algorithms are typically validated extensively in simulation, while only limited real-world trials have been performed.

Chapter 3

Cantilever Construction

In order for a group of robots to self-assemble into a bridge to enable them to cross a gap, the structure must initially be built from only one side, creating a cantilever. A simple analysis highlights the difficulty of designing strong cantilever structures. Consider the cantilever in Figure 3.1, consisting of a beam of length L under a *uniformly distributed load* (UDL) of magnitude w per unit length. Examining static equilibrium reveals that the maximum shear force and moment in the cantilever will occur at the support, denoted $S_{s,\vdash}$ and $M_{s,\vdash}$ respectively. They are given by:

$$S_{s,\vdash} = wL \tag{3.1}$$

$$M_{s,\vdash} = \frac{wL^2}{2} \tag{3.2}$$

The deflection at the tip Δ can also be calculated. For a beam made of material with Young's modulus E and constant second moment of area along its length I, this can be taken from standard formulae [182] to be:

$$\Delta = \frac{wL^4}{8EI} \tag{3.3}$$

These equations highlight the difficulties in designing long cantilevers. High values of shear force and moment will lead to high internal stresses. In order for the structure to not collapse, these stresses must be kept below certain levels, but since $M_{s,\vdash} \propto L^2$ this quickly becomes challenging to achieve as length increases. Engineers therefore typically carefully design cantilevers to be able to safely withstand significantly higher moments as the span increases. Compensating for deflection is even more challenging, as $\Delta \propto L^4$. High deflections could cause the cantilever to not meet a target location on the other side of the gap.

This chapter considers how a group of robots can self-assemble into a cantilever spanning as far as possible across a chasm. These cantilevers should be as long as possible while not collapsing: as demonstrated above, keeping this from happening becomes significantly harder as they grow in length. Force-aware self-assembly algorithms are developed, where agents use measurements of forces within the structure to inform the choice of where they should contribute to it themselves.

This chapter is organised as follows. Section 3.1 formally defines the problem of cantilever construction considered in this chapter. Section 3.2 introduces the simulation

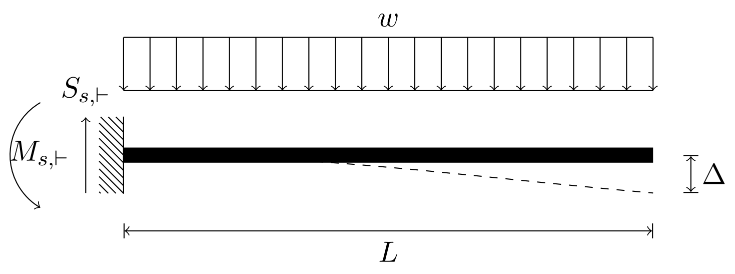


Figure 3.1. A simple cantilever of length L, loaded with a UDL of magnitude w per unit horizontal length. This loading causes a shear force and moment at the support, denoted $S_{s,\vdash}$ and $M_{s,\vdash}$ respectively, as well as a deflection at the tip Δ.

environment used to develop and validate the distributed force-aware algorithms that address this problem. In Section 3.3, optimal cantilevers are computed to compare the performance of the algorithms to. These algorithms are described in Section 3.4: an algorithm in which only one agent can move at a time is introduced first, which has two variants. Following this, a second algorithm that allows for multiple agents to move at once is presented. Section 3.5 describes the simulations performed to verify these algorithms, and shows how well they perform under a variety of test conditions. Finally, the chapter is summarised in Section 3.6.

The work on this chapter is based on the author's published work [183], with certain aspects expanded upon.

3.1 Problem Formulation

This chapter considers a homogeneous group of N robotic agents that self-assemble into cantilevers as shown in Figure 3.2. Agents are squares of side length l and weight W that reside in a 2D grid for simplicity. Positions within this grid are referred to in $(row, column)$ format, with row number increasing downwards and column number increasing to the right. A *fixed support surface* occupies positions $\{(r, c) \mid r > 0 \wedge c = 0\}$, from which the structure extends rightwards.

A continuous supply of agents is available at position $(0, 0)$ which must find their place in the structure. Agents are initially *active*, and become *placed* when they have reached a suitable location to place in. Placed agents cannot move, and are not able to become active again. Active agents can advance one step along the perimeter of the structure in each *timestep* using the 2D sliding cube model illustrated in Figures 2.8a – 2.8b: they can move into an empty space in their Moore neighbourhood as long as this space is in the von Neumann neighbourhood of another agent or the fixed support. These motions are illustrated in Figure 3.2. All agents can communicate with others in their von Neumann neighbourhood.

Cantilevers are restricted to have *continuous* rows and columns of placed agents. This is defined to mean that each column must be filled from row 1 without any gaps, and $n_c \geq n_{c+1}$, where n_x denotes the number of agents in column x. This condition ensures the resulting structures get progressively thicker towards the root, acting as a brace to transfer the weight of the structure onto the fixed support.

Agents are assumed to be able to connect to each other on any face, and also to any

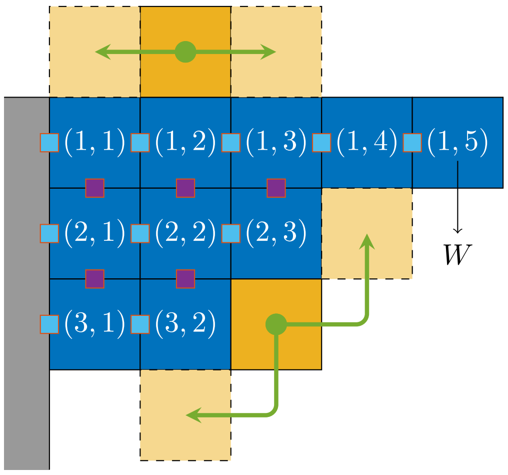


Figure 3.2. An example cantilever consisting of square self-assembling robotic agents (blue) connected to a fixed support (grey). The numbers show the agent location coordinate system. Each agent has a self-weight W, only shown once for simplicity. Row and column links are shown in cyan and purple respectively, each with orange borders. Active agents are shown in yellow with their links hidden, highlighting how they can move along the same row or between adjacent rows by following the paths shown in green.

point of the fixed support. These connections are termed *links*: links along the same row and column are called *row* and *column* links respectively. The *links of column c* are defined as the row links on the left of column c, and the column links within column c.

Each agent has the ability to sense the moment M and axial force F in the links on each of its four faces. These are compared to allowable limits $M_{allowable}$ and $F_{allowable}$ respectively to calculate the *criticalness* γ of each link as:

$$\gamma = \max\left(\gamma^M, \gamma^F\right) \tag{3.4}$$

where:

$$\gamma^M = \frac{|M|}{M_{allowable}} \tag{3.5}$$

$$\gamma^F = \frac{\max(F,0)}{F_{allowable}} \tag{3.6}$$

γ^M and γ^F are referred to as the *moment* and *axial force criticalness* respectively. Each combination of $M_{allowable}$ and $F_{allowable}$ is referred to as a *limit pair*. Links with $\gamma \geq 1$ are described as *critical* and are deemed to be near failure, but not necessarily immediately about to fail. This is analogous to how civil engineers consider safe stress levels in a building to be considerably lower than the failure strength of the constituent materials. If no links are critical, the structure is defined as *stable*, else it is *unstable*.

It should be noted that (3.5) implies links are strong to moments in all directions, while (3.6) implies links are strong in compression but weak in tension. It is also assumed by (3.4) that the shear strength of links is large in comparison to the axial and moment capacity, thus it is not included in this equation. These criteria are chosen to emulate existing modular robots such as SMORES which also have a high strength in

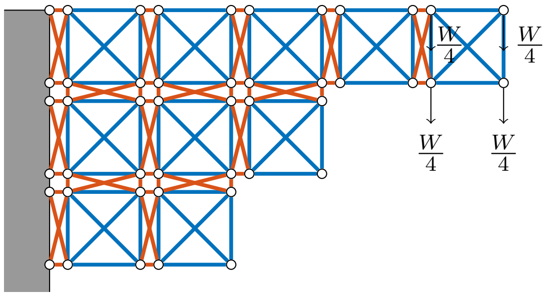


Figure 3.3. The conversion of the cantilever in Figure 3.2 to a truss. Agents are shown as blue members, and links as orange members. A weight of $\frac{W}{4}$ acts downwards from each agent node (only shown for one agent). Unmade links are hidden. © 2021 IEEE.

shear compared to in bending and axially [87]. They are also similar to simple mechanical connections such as the press stud, and so serve as a good approximation of a range of potential mechanical connection mechanisms. While this chapter considers only M and F for simplicity, the exact failure criteria of a real robotic system will be unique to the mechanical design of its linkages. Indeed the hardware developed in Chapter 6 uses a different metric more suitable to the design of its connectors. Nevertheless, using M and F in this chapter allows a generalised form of the problem to be studied.

The *length* of the cantilever L is the number of columns it contains. The objective of the algorithms in this chapter is to self-assemble cantilevers that are as long as possible, while aiming to correct any unstable configurations that occur during construction as soon as they arise.

3.2 Simulation Environment

A suitable simulation environment was required to develop these algorithms. This simulator should be able to calculate M and F for any configuration of agents quickly and accurately. Existing simulators, including Webots [184] and CoppeliaSim [185] were considered, but they do not allow for easy calculation of forces at specific locations within solid bodies. For this reason, a custom simulator was required.

The simulator was written in Python, as it is a powerful cross-platform language with a large variety of packages to enable different functionalities. In order to calculate M and F, the structure is modelled as a 2D truss as shown in Figure 3.3, a method inspired by Brodbeck and Iida [16]. The forces within truss members are calculated using the anaStruct module for Python [186], and converted into a value of M and F for each link. A side length of 0.1 m is used to reflect an estimate of existing modular robotic platforms [25, 60, 74, 87]. The weight is conservatively calculated as if each agent were a solid cube of aluminium to give $W = 19.3\,\mathrm{N}$, significantly greater than existing modular robots.

While the simulator is able to quickly calculate the static forces within the structure, it does have certain limitations. Firstly, restricting the environment to a grid simplifies it considerably, and reduces the range of robotic platforms that the developed self-assembly algorithms could be applied on. However, as seen in Section 2.2.1.B, lattice-type designs of modular robot are popular, so a wide range of systems could potentially utilise these algorithms. The deflection of the structure is also not considered, which could become significant for large structures, preventing agents from fitting into the desired locations in the grid: the simulator assumes that either the agents are sufficiently rigid that this will not be an issue, or the method agents use to connect to one another is able to correct for such misalignments. A further limitation is that dynamic forces as agents move around the structure are neglected. This could be reflected in the real-world by moving agents slowly so as to not generate large impulses during actuation. Despite these limitations, the simulator is suitable for the development of the self-assembly algorithms contained in this thesis, and improvements could be made in future to address these concerns.

More details of the simulator are given in Appendix A. This includes a brief discussion of other methods of calculating M and F that were explored before choosing the truss approximation method.

3.3 Offline Structural Optimisation

Optimal cantilevers were calculated offline to measure the performance of the self-assembly algorithms against. A configuration of N agents and length L bodylengths is considered optimal if it is stable and no other stable configuration of N agents exists that has length $> L$. There may be multiple optimal configurations of N agents. In this section, the optimisation procedure is first described, before the resulting optimal cantilevers are presented.

3.3.1 Optimisation Procedure

To find optimal cantilevers, the concept of *integer partitioning* from number theory is used [187]. A cantilever configuration of length L bodylengths may be represented by the number of agents in each column as $[n_1, n_2, ..., n_L]$, hence $\sum_c n_c = N$, and this vector is a partition of N. Since rows and columns are assumed to be continuous, this vector uniquely describes a cantilever configuration. For example, the structure in Figure 3.2 can be represented as $[3, 3, 2, 1, 1]$, and this is the only valid structure for this partition.

The number of partitions for a given N scales with $\mathcal{O}\left(e^{\sqrt{N}}\right)$. To allow tractable computation, a different procedure is therefore used for small ($N \leq 50$) and large ($N > 50$) structures. These procedures are described below.

3.3.1.A Small Structures

For small structures, it is possible to list all the possible cantilever configurations within a reasonable timeframe. Each partition of $N \leq 50$ was calculated then modelled to find the maximum M and F in all links. A limit pair was then specified and these precomputed data analysed to find the configurations that result in the longest stable structures of N agents for all $N \leq 50$ for this limit pair. This search is exhaustive, so optimality is guaranteed.

3.3.1.B Large Structures

For large structures, listing all possible configurations is intractable, so a different procedure is used for $N > 50$. The number of configurations that need to be modelled is reduced by observing that the optimal arrangement of agents at the tip remains constant as N and L increase.

The optimisation procedure for large cantilevers begins by specifying a limit pair. A number of agents N_{test} is then chosen that is believed might be able to build a stable cantilever $L_{max}+1$ agents long, where L_{max} is the length of the longest optimal configuration that has been previously found for this limit pair. Exploiting the observation about the optimal configurations at the tip, all known optimal configurations of $N < N_{test}$ and $L > (L_{max} - 5)$ are first considered. The numbers of agents in each column of each stable configuration are compared to find the longest portion at the tip that is common across all these configurations. All configurations of this N_{test} are then enumerated, as for small structures. However, those that do not include this shape at the tip are discarded.

Once the reduced list of configurations has been generated, each one is modelled and any stable configurations of the specified N_{test} are saved. The procedure is repeated with different N_{test} until the minimum number of agents required to build a stable structure of length $L_{max} + 1$ bodylengths containing the precomputed tip configuration with links of the given strength is found.

Exhaustive search again ensures these cantilevers are optimal, but only under the given assumption about tip configuration. A verification process is then used to prove that these structures are indeed optimal. For each pair of N_{test} and L believed to be optimal, all configurations of $N_{test} - 1$ agents and length L bodylengths are enumerated and modelled to find the maximum M and F in links within them. If any stable configurations are found, then the assumption about the optimal arrangement of agents at the tip was incorrect in this case, as the same L could be reached with smaller N than in the calculated optimal structures. All structures of $N_{test} - a$ agents and length L bodylengths are then enumerated and tested, beginning with $a = 2$ and continuing with incrementing a until the lowest number of agents required to build a stable structure this length is found. Many iterations of this procedure with increasingly large a would be computationally expensive, but the assumption about the configuration at the tip gives a good value of N_{test} to use during this iteration.

3.3.2 Limit Pairs

Optimal structures were calculated for multiple limit pairs to examine how they vary in a range of scenarios, and to allow the flexibility of the self-assembly algorithms to be investigated. Each pair is generated by choosing a number of agents N_g, arranging $N_g + 1$ agents in a cantilever of a single row, and measuring M at the root: this will be the largest M across all the links. It is chosen to set $M_{allowable}$ to be 10 % less than this value, thus a single-thickness cantilever of length N_g is stable, but close to instability. A single-thickness cantilever has $F = 0$ at the root, thus cannot be used to calculate $F_{allowable}$. Instead, this is chosen to be $10N_gW$.

These limits reflect how certain connections are better able to hold structures vertically than horizontally. For example, the SMORES robot [87] can hang an average of 11 agents vertically from a single connector without failure, but can only support a cantilever of 3 agents.

Three limit pairs were chosen for which to calculate optimal structures and test the algorithms. These are given in Table 3.1, and are described as *weak*, *medium*, and *strong*.

Name	N_g	$M_{allowable}$ (Nm)	$F_{allowable}$ (N)
Weak	3	13.9	579
Medium	6	42.6	1159
Strong	9	86.9	1738

Table 3.1. The limit pairs the self-assembly algorithms were tested with.

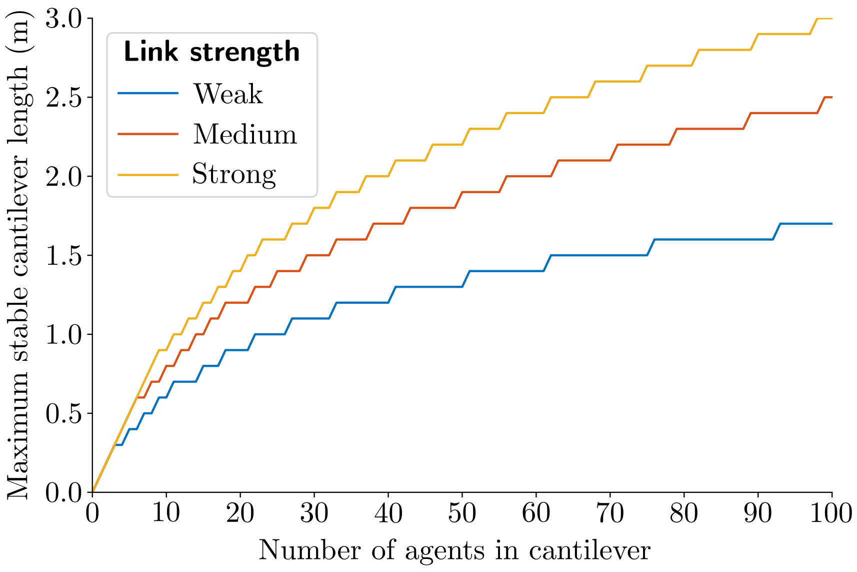


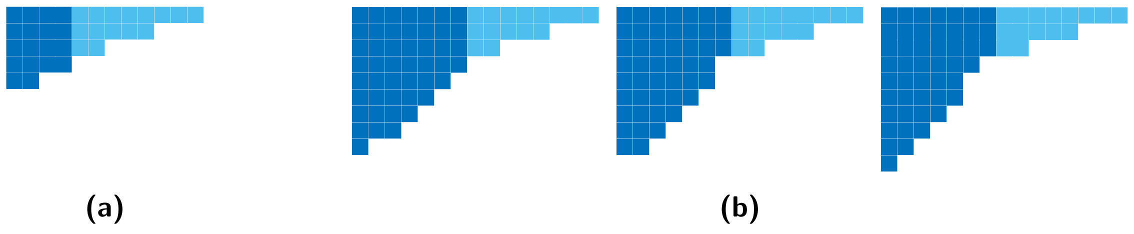
Figure 3.4. The number of agents required to build optimal cantilevers of varying lengths.

3.3.3 Resulting Optimal Cantilevers

The optimisation was carried out for $N \leq 100$ agents for each of the three limit pairs. The results are summarised in Figure 3.4. It was found that higher allowable limits lead to longer possible stable structures, and that the rate of increase in L slows as agents are added.

Tables containing complete lists of all the optimal cantilevers are given in Appendix B, but Figure 3.5 shows examples of them, chosen as the minimum N that can reach the specified L. Weak links are used in Figures 3.5a & 3.5b, which results in the construction of cantilevers that are short and thick. There is only one optimal configuration for $L = 12$, $N = 33$, but three for $L = 15$, $N = 62$. Figures 3.5c & 3.5d show structures with strong links, where fewer agents are required to reach equivalent lengths as more slender cantilevers are possible. Note also how the portions at the tip of structures with the same link strength are the same, as observed in Section 3.3.1.

The verification process revealed one instance where the assumption about the tip configuration was invalid. For strong links and $L = 30$ bodylengths, it was predicted that the first 6 rows would have lengths $[30, 21, 14, 11, 8, 7]$. The optimum cantilevers that satisfy this condition consist of $N = 99$ agents. However, the verification process revealed structures of $N = 98$ agents with top row lengths of $[30, 21, 14, 11, 8, \underline{6}]$. No stable structures exist for $L = 30$ bodylengths, $N = 97$ agents, thus the verification process stops when $a = 2$.



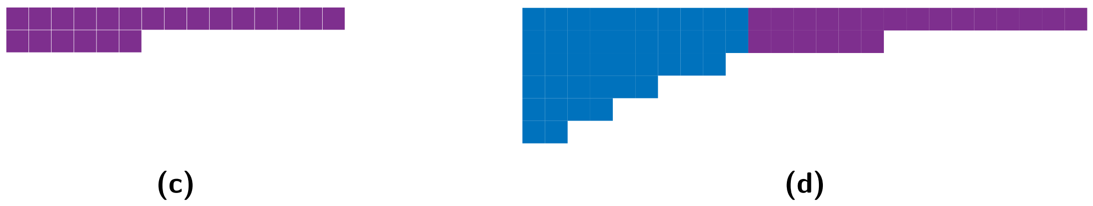


Figure 3.5. Optimal cantilevers with a given number of agents for different limit pairs. **(a)** A 1.2 m cantilever with weak links requires 33 agents. **(b)** Increasing length to 1.5 m requires 62 agents and results in three stable configurations as shown. **(c)** Strong links can reach 1.5 m with only 21 agents. **(d)** An optimal structure of strong links reaching a length of 2.5 m, which requires 62 agents. The common portion at the tip for weak links is shown in cyan, and for strong links in purple. © 2021 IEEE.

3.4 Algorithm Design

The computation of optimal cantilevers in the previous section gives a target for the structures that the agents should be able to self-assemble into. This section describes the self-assembly algorithms the agents follow, with a discussion of the theoretical basis behind them. Two algorithms are presented, called the *sequential* and *parallel* algorithms. Both algorithms begin with an agent placed in position $(1, 1)$. In the sequential algorithm, new active agents are released only once the previous one has placed, whereas the parallel algorithm allows for multiple active agents concurrently. Both algorithms incorporate measurements of M and F within each link to construct stable structures, and run in a distributed manner on each agent. Active agents keep track of their movements to calculate their positions in the coordinate system.

3.4.1 Theoretical Basis

Here, a result from structural mechanics which influences the design of the self-assembly algorithms is introduced.

Theorem 1. *Consider a cantilever of length L whose height $h(x)$ is a continuous and monotonically decreasing function of the distance x from a fixed support, as shown in Figure 3.6. The cantilever has constant breadth b into the page, and the only load is a UDL of magnitude w per unit horizontal length. For such a cantilever, increasing the height $h(x)$ at a distance $x = x_0$ from the support causes a decrease in the maximum longitudinal stress experienced in the cross-section here, $\sigma_{max}(x_0)$.*

Proof. Such a cantilever can be analysed using elastic beam theory [188]. A cut through the cross-section is made at x (Figure 3.6b) to reveal the internal moment $\tilde{M}(x)$. This

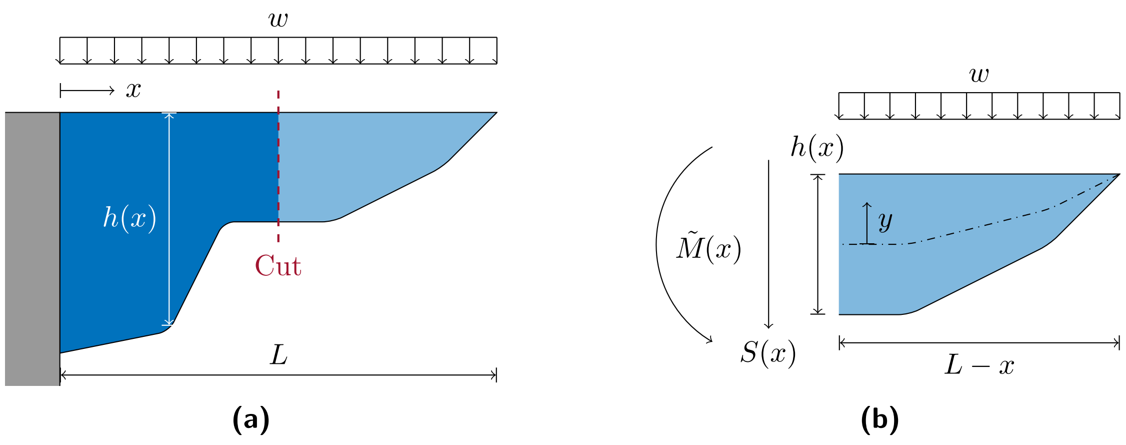


Figure 3.6. **(a)** Profile of a cantilever (blue) with continually-varying height $h(x)$ a distance x from the root extending from a fixed support (grey). The only load is a UDL of magnitude w per unit horizontal length. **(b)** A cut through this cantilever at x, showing the internal shear force $S(x)$ and moment $\tilde{M}(x)$ on the exposed cross-section. The dashed line shows the neutral axis, which is assumed to be in the centroid of the cross-section. © 2021 IEEE.

can be found from static equilibrium to be:

$$\tilde{M}(x) = \frac{w(L-x)^2}{2} \tag{3.7}$$

For each x, $\sigma_{max}(x)$ occurs where the distance y from the neutral axis (assumed to be at the centroid of the cross-section) is as large as possible. This can therefore be calculated as:

$$\begin{aligned}
\sigma_{max}(x) &= \frac{\tilde{M}(x)y}{I(x)} \\
\sigma_{max}(x) &= \frac{\frac{w}{2}(L-x)^2 \cdot \frac{h(x)}{2}}{\frac{bh(x)^3}{12}} \\
\sigma_{max}(x) & \frac{3w(L-x)^2}{bh(x)^2}
\end{aligned} \tag{3.8}$$

where $I(x)$ is the second moment of area of the cross-section, obtained from a standard formula [188]. This equation shows that $\sigma_{max}(x_0)$ decreases as $h(x_0)$ increases. □

These equations are derived for beams where the height is a continuous function along its length and the only load is a UDL. In contrast, the scenario considered in this chapter (defined in Section 3.1) concerns cantilevers made of discrete agents under the self-weight of each agent. The exact equations are therefore not applicable, although the trends are used to inform the design of the self-assembly algorithms.

3.4.2 Sequential Algorithm

In the sequential self-assembly algorithm, the agents in the structure sample their sensor readings once an active agent places itself, and hold these values to communicate

Algorithm 3.1. The message-passing variant of the sequential cantilever construction algorithm.

```
1  Initialise at $(0,0)$;
2  while $row = 0$ and $column < L$ do
       // Active (gathering mode)
3      Make one step right;
4      Record $M$ and $F$ from agent below into $\gamma_c^{\alpha,\beta}$;
5  if No link is critical then
       // Active (placing mode, to extend)
6      Place at tip;
7  else
       // Active (placing mode, to reinforce)
8      Calculate $p_{column}(c)$ from $\boldsymbol{\gamma}^{\alpha,\beta}$;
9      while Not placed do
10         $c_{target} \leftarrow$ sample from $p_{column}(c)$ without replacement;
11         Move to column $c_{target}$;
12         if Valid location then
13             Place here;
14 while True do
       // Placed
15     $M_l, F_l, M_b, F_b \leftarrow M$ & $F$ in left & bottom links;
16     for $\alpha$ in $[M,F]$ and $\xi$ in $[l,b]$ do
17         Send upwards $\max(\alpha_{\beta,self}, \alpha_{\xi,agent\ beneath})$;
```

to the next active agent. This effectively means that the weight of the active agent is not included in the measurements of M and F. Two variants of the sequential algorithm are developed, called the *message-passing* and *local* variants. They differ only in how the active agent receives the force information from the placed agents.

3.4.2.A Message-passing Variant

The message-passing variant is shown in Algorithm 3.1. Lines 2 – 13 detail the operation of the agent while it is active. During this stage, it moves between two *modes*. In the first mode, `gathering`, it travels to the tip of the cantilever, receiving data from the agents in row 1. These data contain the maximum M and F observed for any links belonging to the corresponding columns, obtained by passing messages between placed agents in a manner that will be explained in due course below. The active agent creates four arrays $\left\{\boldsymbol{\gamma}^{\alpha,\beta}\ \forall\ \alpha \in \{M,F\} \wedge \beta \in \{row, column\}\right\}$ for the measurements of M and F in row and column links respectively. These arrays are arranged such that the c^{th} component of the appropriate array contains the maximum γ^M or γ^F in row or column links belonging to column c: this component is denoted $\gamma_c^{\alpha,\beta}$.

When the tip is reached, the active agent switches to the `placing` mode. The algorithm aims to construct cantilevers that are as long as possible, thus the active agent places at

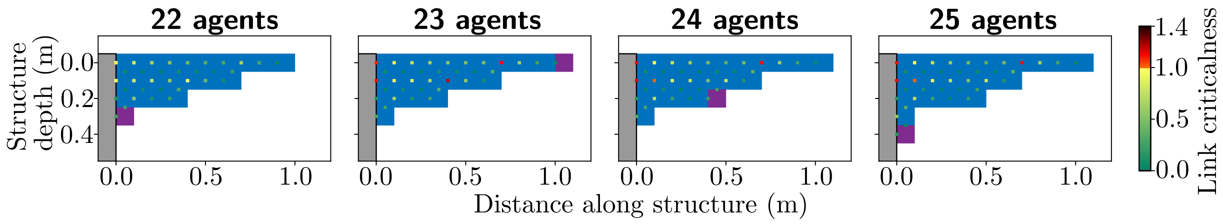


Figure 3.7. Frames from a cantilever construction sequence for weak links when the column with the most critical link is chosen to reinforce. The agents (blue) extend from the fixed support (grey). The most recently placed agent in each frame is shown in purple, and links are coloured by their criticalness.

the tip if it believes the structure is stable based on the force information it has received (Line 6). Otherwise, it chooses a column to reinforce. Agents reinforce the bottom of columns in an effort to better transfer the load onto the fixed support.

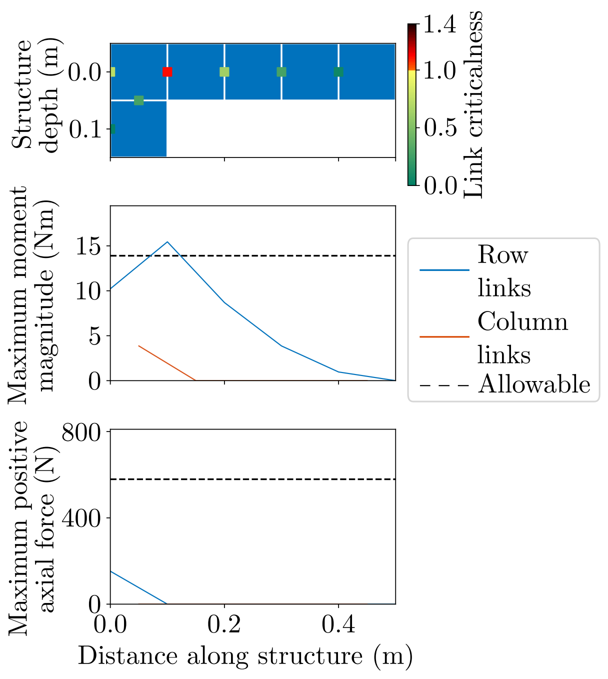
Placing in a column will increase the height of this column, which according to Theorem 1 causes a reduction in the maximum longitudinal stress on its left-hand face. This suggests that a good approach could be to add to the bottom of columns that have high $\gamma^{\alpha,row}$ such that this reduced longitudinal stress also causes a reduction in M and F. However, the case analysed in Theorem 1 is that of a cantilever of continually varying height under a UDL, so this cannot be used directly in the situation considered here where the cantilever comprises discrete agents with their own self-weights. Indeed, it was found that always increasing the thickness of the column containing the link of highest γ quickly led to situations where the algorithm would repeatedly place in the same column without any benefit to the stability of the structure.

Figure 3.7 gives an example of this naïve approach for weak links. A stable structure of $L = 10$ bodylengths is successfully built with $N = 22$ agents. This can therefore be extended by the next agent ($N = 23$), which makes the structure unstable. The subsequent agent ($N = 24$) places in column 5 to provide reinforcement to the link in the structure with the highest γ, which belongs to this column. The highest γ then occurs in the row link at the top of column 1, but placing agents in this column does not reduce γ in this link. The construction therefore enters a livelock, where agents repeatedly place in this column instead of a more suitable locations, such as columns 2 or 8.

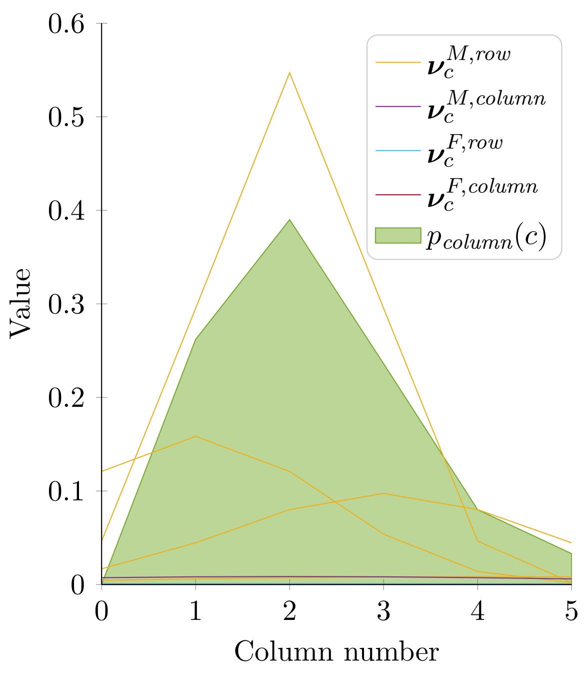
In order to employ the basic result of Theorem 1 in the scenario of discrete agents each with their own self-weight, a probabilistic approach was taken. Active agents are more likely to place in columns with links of high γ, but not guaranteed to. This allows them to explore a wider solution space so the construction avoids getting stuck in a livelock, as it did in the example in Figure 3.7. A probability mass function $p_{column}(c)$ describing the probability of placing in column c is calculated from the $\boldsymbol{\gamma}^{\alpha,\beta}$ arrays in Line 8 as follows:

1. The values in $\boldsymbol{\gamma}^{\alpha,column}$ are shifted by one to the right, so that a high γ in column links is reflected by a high probability of reinforcing in the adjacent column, as this will be more effective at reducing γ in these links.

2. For each member $\gamma_c^{\alpha,\beta}$ of each array $\boldsymbol{\gamma}^{\alpha,\beta}$, an *urgency distribution* $\boldsymbol{\nu}_c^{\alpha,\beta}$ is calculated. This is defined as the Gaussian distribution centred on column c with variance $\left(\gamma_c^{\alpha,\beta}\right)^{-2}$, then scaled by a factor of $\left(\gamma_c^{\alpha,\beta}\right)^2$ and sampled for each column in the



(a)



(b)

Figure 3.8. An example of how the probability mass function $p_{column}(c)$ is calculated. **(a)** A cantilever (blue squares) with weak links, coloured by their criticalness, and the values of M and F within its links. **(b)** The urgency distributions $\boldsymbol{\nu}_c^{\alpha,\beta}$ calculated for the M and F in each column c, as well as the resulting $p_{column}(c)$.

cantilever $x \in \mathbb{Z} : 1 \leq x \leq L$. The x^{th} component of $\boldsymbol{\nu}_c^{\alpha,\beta}$ is thus given by:

$$\boldsymbol{\nu}_c^{\alpha,\beta}|_x = (\gamma_c^{\alpha,\beta})^2 \cdot \frac{\gamma_c^{\alpha,\beta}}{\sqrt{2\pi}} \exp\left(-\frac{(\gamma_c^{\alpha,\beta})^2(x-c)^2}{2}\right) \tag{3.9}$$

The area underneath this curve is equal to $\left(\gamma_c^{\alpha,\beta}\right)^2$, so this spreads the impact of the critical links in column c across the whole cantilever, but priorities the region around column c for reinforcement the most. It is chosen to square $\gamma_c^{\alpha,\beta}$ to magnify the effect of critical links, for which $\gamma_c^{\alpha,\beta} \geq 1$.

3. For a cantilever of length L, there will be $4L$ urgency distributions. These are summed to calculate the combined urgency distribution $\hat{\boldsymbol{\nu}}$ as:

$$\hat{\boldsymbol{\nu}} = \sum_{\alpha \in \{M,F\}} \sum_{\beta \in \{row,column\}} \sum_{c=1}^{L} \boldsymbol{\nu}_c^{\alpha,\beta} \tag{3.10}$$

4. Finally, $p_{column}(c)$ is calculated by dividing each component of $\hat{\boldsymbol{\nu}}$ by the sum of all the components.

Example urgency distributions and the resulting probability mass function for a simple cantilever are shown in Figure 3.8. The cantilever in Figure 3.8a has a critical row link in column 2. The urgency distributions calculated by the next active agent are shown in

Figure 3.8b, which are combined to give $p_{column}(c)$ as shown. The probability of placement is highest in column 2, but also reasonably high in the adjacent columns 1 and 3. Also note that only columns 1 and 2 are valid placement locations due to the continuity condition.

This probability mass function is sampled from without replacement in Line 10 to produce a target column c_{target}. The active agent travels to the bottom of this column, and if this placement is locally determined to be valid (meaning row and column continuity is maintained) it places here. If not, another sample is drawn from the distribution. This repeats until a valid location is chosen. Column 1 is always valid, thus the `while` loop terminates after at most L iterations.

Once the active agent has placed itself, it remains stationary and runs the message-passing procedure in Lines 14 – 17. Each agent receives the M and F that the agent below measures in links on its own bottom and left faces, or larger equivalent values from an agent below that one. These measurements are compared to measurements in the equivalent links of the agent in question, and the maximum values are passed to the agent above. If any link is not currently made it reads a value of zero. The agent in the top row thus receives the maximum M and F in all row and column links belonging to this column. These messages are passed after one active agent places itself and before another initialises, therefore the weight of the active agent is excluded.

3.4.2.B Local Variant

The message-passing variant requires coordination between agents that have been placed to transmit information within the structure. It requires a large number of communications before each active agent can place itself, which increases the energy consumption and number of opportunities for a communication failure to occur: communicating the information to the n^{th} agent requires each placed agent to communicate its force measurements to the agent above, so the total number of communications required to place N agents, given the symbol ϕ, scales according to $\phi_{message\text{-}passing} = \mathcal{O}(N^2)$.

The local variant was developed to reduce the amount of inter-agent communication required by incorporating more knowledge from Section 3.4.1. For the beam considered in Theorem 1, $\sigma_{max}(x)$ occurs on the top and bottom of the cross-section as $|y|$ is greatest here. In the case of a beam of discrete agents with their own self-weights, it can therefore be predicted that the maximum M and F within links of a given column is likely to occur in a link connected to an agent on the edge of the cantilever. This means that the active agent can reasonably approximate how the maximum M and F vary along the length of the structure by only receiving the measurements of these *exterior* agents, without the need for passing messages.

The local variant of the sequential algorithm is shown in Algorithm 3.2: it is very similar to the message-passing variant, so the differences are emphasised. In the `gathering` mode (Lines 2 – 4), the active agent must traverse the full perimeter of the beam and communicate with all exterior agents. It compares the M and F values measured by agents at the top and bottom of each column and saves the maximum ones to $\gamma^{\alpha,\beta}$. This makes the message-passing behaviour in Lines 14 – 17 obsolete, but requires the active agent to traverse a longer distance to obtain the necessary data. The local variant is important as it validates the assumption that only the sensors of exterior agents are salient, and is built upon in the parallel algorithm.

In the local variant, agents continue to be required to coordinate to sample their

Algorithm 3.2. The local variant of the sequential cantilever construction algorithm, emphasising differences with the message-passing variant (Algorithm 3.1).

```
1  Initialise at $(0, 0)$;
2  while not ($row > 0$ and $column = 1$) do
       // Active (gathering mode)
3      Make one step clockwise;
4      Record $M$ and $F$ from agent below or above into $\gamma_c^{\alpha,\beta}$;
5  if No link is critical then
       // Active (placing mode, to extend)
6      Place at tip;
7  else
       // Active (placing mode, to reinforce)
8      Calculate $p_{column}(c)$ from $\gamma^{\alpha,\beta}$;
9      while Not placed do
10         $c_{target} \leftarrow$ sample from $p_{column}(c)$ without replacement;
11         Move to column $c_{target}$;
12         if Valid location then
13             Place here;
14 ~~while True do ...~~
```

sensors once an active agent has placed itself, then to hold this value to communicate with the next agent. It would be possible to read the sensors as the active agent traverses instead to further reduce the coordination required, but this is not done so that the two variants are more comparable. Excluding this additional communication, the n^{th} agent to place only requires $2L_n$ communications, where L_n is the length of the structure to which agent n is added. In the worst case, a structure consisting of a single row of agents, $L_n = n - 1$ so $\phi_{local} = \mathcal{O}(N^2) = \phi_{message\text{-}passing}$. However, structures will eventually require reinforcement so typically $L_n < n$; this difference increases as more reinforcement is required, so the theoretical worst-case limit will rarely be reached.

3.4.3 Parallel Algorithm

The sequential algorithm has two drawbacks. Firstly, construction takes a long time as only one agent is active at once. Secondly, the 'sample and hold' procedure used in communicating the force information to the active agent increases the inter-agent coordination required and the number of messages that must be reliably sent.

In the parallel algorithm, agents follow a procedure based upon the local variant of the sequential algorithm, but with allowances made to enable multiple agents to move at once. Active agents are released at fixed intervals of δ timesteps. If the initialisation position is still occupied, they are released at the next possible timestep. The agents *advance* in a random order each timestep to simulate synchronisation inconsistencies in a real system. It is assumed that sensing and communication is instantaneous.

The procedure agents use when it is their turn to advance is presented in Algorithm 3.3. Each active agent now transitions through *three* modes as they find their place within

Algorithm 3.3. The parallel cantilever construction algorithm.

```
1  switch mode do
2      case gathering do
3          Record $M$ and $F$ from placed agent above or below into $\gamma_c^{\alpha,\beta}$;
4          if $row > 0$ and $column = 1$ then
5              Calculate $p_{column}(c)$ from $\boldsymbol{\gamma}^{\alpha,\beta}$;
6              $c_{target} \leftarrow$ sample from $p_{column}$ without replacement;
7              mode ← placing;
8          else
9              Make step*;
10     case placing do
11         if In column $c_{target}$ then
12             Attempt placement;
13             if Placement succeeded then
14                 Agent no longer active;
15                 return
16             else
17                 $c_{target} \leftarrow$ sample from $p_{column}(c)$ without replacement;
18         Make step*;
19     case swapping do
20         mode ← previous mode;
21 if Agent stationary for $> \delta$ timesteps then
22     Attempt placement;
23     if Placement succeeded then
24         Agent no longer active;
```
* Steps made if the agent is not the sole support of another

the structure. On initialisation, they are in the `gathering` mode (Lines 2 – 9), where they move around the perimeter of the structure in the same manner as the local variant of the sequential algorithm. As they move, they communicate with exterior agents to obtain the maximum M and F currently experienced by their links, as opposed to a value sampled and held from an earlier timestep. When a `gathering` active agent begins advancing below row 0 and in column 1, it calculates the probability mass function $p_{column}(c)$ in the same way as the sequential algorithm, samples from it without replacement to set a target column c_{target}, and transitions to the `placing` mode.

In the `placing` mode (Lines 10 – 18), active agents first check if they have reached c_{target}. If they have, they attempt to place here. If this placement is valid, the agent places here (Line 14). It is no longer active, and instead waits to communicate the force information it measures in its links to passing active agents.

In the event of two agents attempting to occupy the same location, they communicate internal states and 'become' one another. For two agents to swap in this manner requires that each remains stationary for one timestep, modelling the time cost this communication

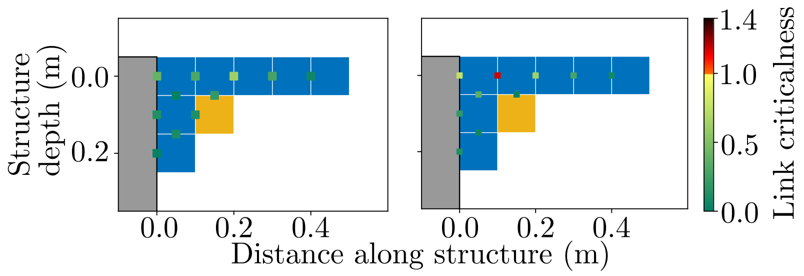


Figure 3.9. An example of how the active agent (yellow) connecting to placed agents (blue) extending from a fixed support (grey) on all adjacent faces hides critical links. Links are coloured by criticalness, assuming weak links. On the left, the critical link is hidden as the active agent provides additional support by connecting on its left face.

would have in real life: in the next timestep, each agent is therefore in the `swapping` mode (Lines 19 – 20), and does not move. There are a small number of exceptions, as described in Section 3.4.3.B.

To avoid active agents blocking each other from moving, they attempt to place if they have been stationary for too many timesteps after advancing (Lines 21 – 24). The number of timesteps before this timeout is denoted δ.

3.4.3.A Active Links

In the sequential algorithm, M and F are only updated after agents place themselves. In contrast, the parallel algorithm calculates this at the end of every timestep, thus the weight of active agents is measured. However, if active agents attach to all adjacent contact surfaces they could reinforce the structure before their location is finalised and thus hide critical links. This effect is illustrated in Figure 3.9: if the active agent connects to the placed agent in $(2, 1)$, the critical row link to the left of $(1, 2)$ is hidden. The active agent will therefore not measure this link as critical, and will not place itself in a suitable location. When only connected by the link on its top face, the active agent will correctly be informed of this critical link.

To prevent active agents from providing reinforcement to the structure, each can only make one link to the fixed portion of the cantilever. Each active agent is restricted to only control one of its four possible links at a time, referred to as its *active link*. By default, the active link is set to be on the lower face of each active agent if the agent is above row 1, and on the upper face otherwise. There are four exceptions to this rule where the active link is set to the left face instead, as shown in Figure 3.10. If the active agent is above row 1, this only occurs when the agent left of it is also active, and there is no placed agent beneath (green agent). If the active agent is below or in row 1, this occurs in three scenarios:

(i) There is no agent above (red agent).

(ii) There is an active agent above, and there is a placed agent to the left (purple agents).

(iii) There is an active agent above that is not swapping, there is an active agent to the left, and the active agent in question is moving upwards (cyan agent).

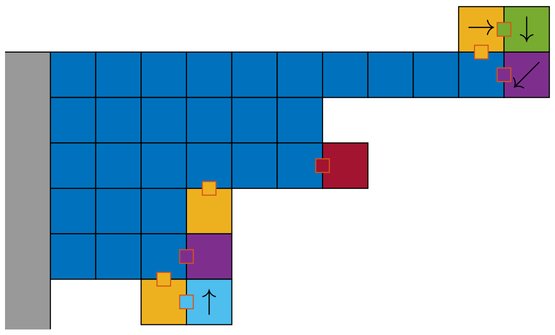


Figure 3.10. Examples of special cases then the active link is set to the left face. Blue agents are `placed` and from a fixed support (grey). Active agents with the default active link (lower face when above row 1, and upper face otherwise) are shown yellow. Agents of other colours are active with an active link on their left face. Links have orange borders and are filled the same colour as the active agent they are controlled by. Arrows show salient directions of motion. © 2021 IEEE.

3.4.3.B Swapping

Two agents travelling in opposite directions that attempt to move into the same space will usually exchange information and 'become' one another. However, there are three-scenarios when this does not happen, as outlined below:

- **Placing:** If the agent that does not initiate the swap is in the `placing` mode and in column c_{target}, it will attempt to place here before swapping. If successful, the agent initiating the swap is stationary in this timestep as a penalty. This allows agents to place as soon as possible, reinforcing the structure and reducing congestion.
- **Direction:** Agents advance in a random order each timestep, so it is possible that two agents travelling in the same direction will attempt to swap with each other. Instead of swapping, the initiating agent will normally abort the swap and remain stationary (Figure 3.11a). However, if the two agents are connected by an active link, the initiating agent will step backwards in the next timestep instead to allow the other agent to vacate this space (Figure 3.11b).
- **Around the tip:** Agents entering the structure are kept separate from those below the structure to allow those below to place before additional agents are added. If an agent in row 1 attempts to swap with one at the tip, it will instead step back in the next timestep to allow the one at the tip to continue (Figure 3.11c).

If the overridden direction of an agent tells it to move into the fixed support, it will instead remain stationary in this timestep.

3.5 Simulation Results

In this section, the simulations performed to evaluate the performance of the cantilever construction algorithms are presented. Due to the stochastic nature of the algorithms, large numbers of trials were required to be performed to observe the average performance.

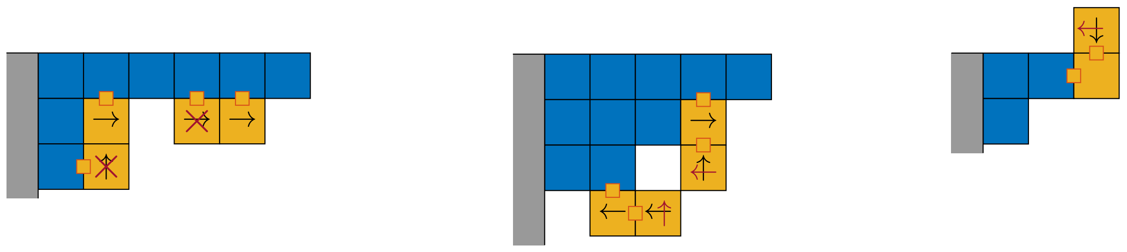


Figure 3.11. Scenarios when active agents should not swap information with each other. Active agents are shown in yellow, placed agents in blue, and the fixed support in grey. Active links are shown as small yellow squares with orange borders. Salient directions of travel are shown by black arrows, and overridden directions in red.

The High Performance Computing facility at The University of Sheffield was used to perform these simulations in a reasonable timeframe. The algorithms were tested for the three limit pairs described in Section 3.3.2.

3.5.1 Sequential Algorithm Performance

Both variants of the sequential algorithm were tested for all three limit pairs up to $N = 100$ agents. A total of 400 trials were performed with each limit pair to accurately evaluate the performance of the stochastic algorithms.

To illustrate how the structures are built by the sequential algorithm, selected frames of the self-assembly sequence generated during one trial of the local variant are shown in Figure 3.12. At $N = 21$ agents, the only critical link is a row link in column 4. This belongs to an exterior agent, so the next active agent receives this information and has a high $p_{column}(4)$. This agent therefore places in this column, making the structure with $N = 22$ agents stable. When $N = 29$ agents, the most critical link in column 3 does not belong to an exterior agent, so the next active agent does not become directly aware of it. However, the link left of the agent at the top of column 2 is critical, so the combination of the urgency distributions causes the next active agent to place in a location that reduces the criticalness of the link in column 3 regardless. At each stage, the top left corner has links of high γ compared to the rest of the structure. This highlights how the root is where M and F are generally greatest, so requires the most reinforcement.

An example structure with the maximum N tested of 100 agents constructed by the message-passing variant for links of medium strength is shown in Figure 3.13. It shows that at this stage γ^M is larger than γ^F in most columns, and that row links experience higher M and F than column links; measurements of all four metrics will, however, influence $p_{column}(c)$. It is possible that the column links could be ignored, thus realising an even simpler control scheme. If this was the case, hardware to deploy these algorithms on could be made less complex, as force sensors would only be required on two faces. Initial simulations showed that the performance impact of ignoring the forces in column links was minimal, but further investigation should be performed to explore this effect in detail.

The average performance for both variants of the sequential algorithm is shown in Figure 3.14. It can be seen from Figure 3.14a that both variants require similar numbers

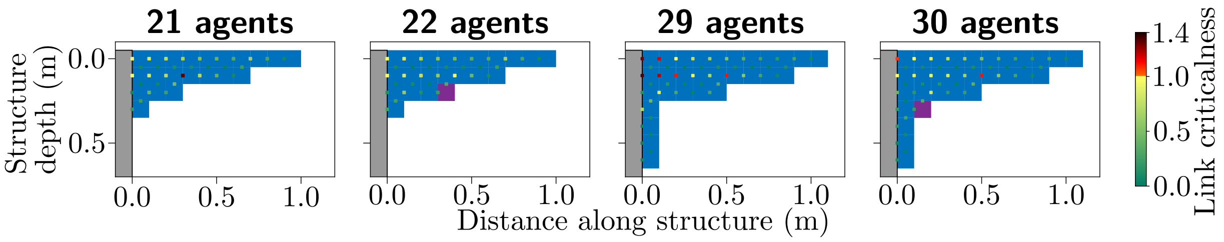


Figure 3.12. Frames from a cantilever construction sequence produced by the local variant of the sequential algorithm for weak links. The agents (blue) extend from the fixed support (grey). The most recently placed agent in each frame is shown in purple when salient, and links are coloured by their criticalness. © 2021 IEEE.

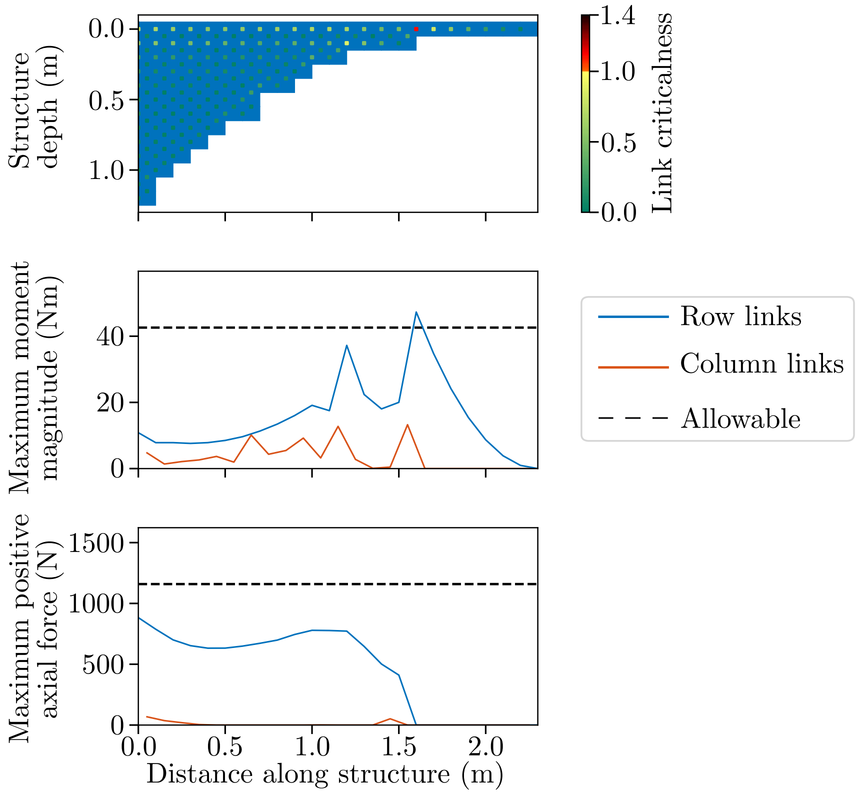
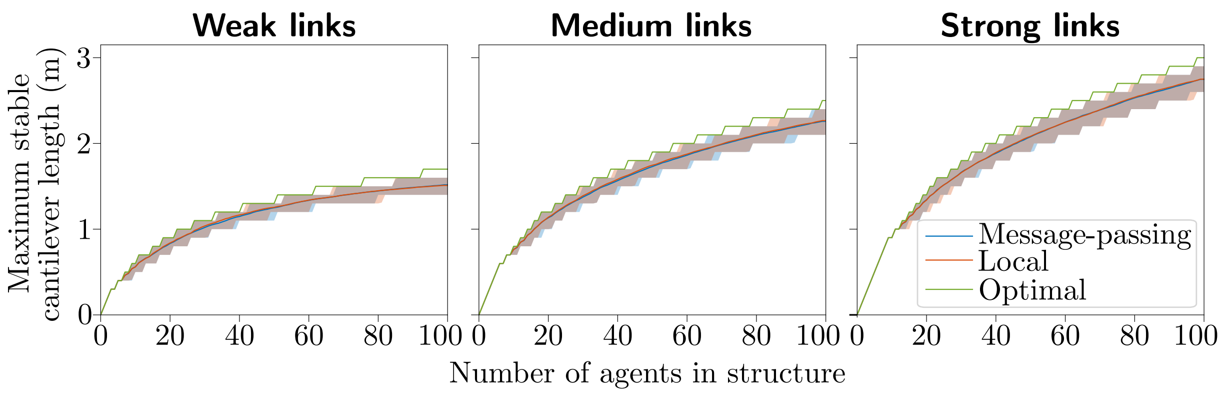
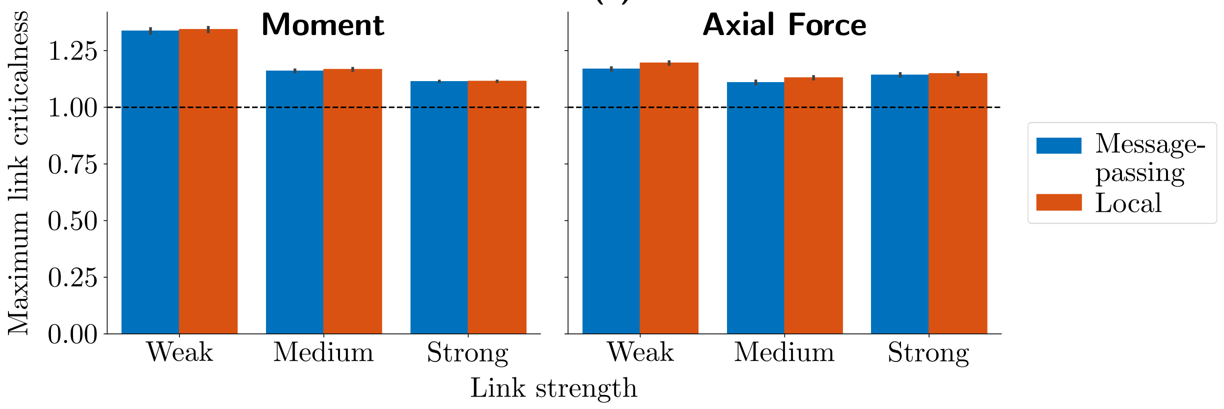


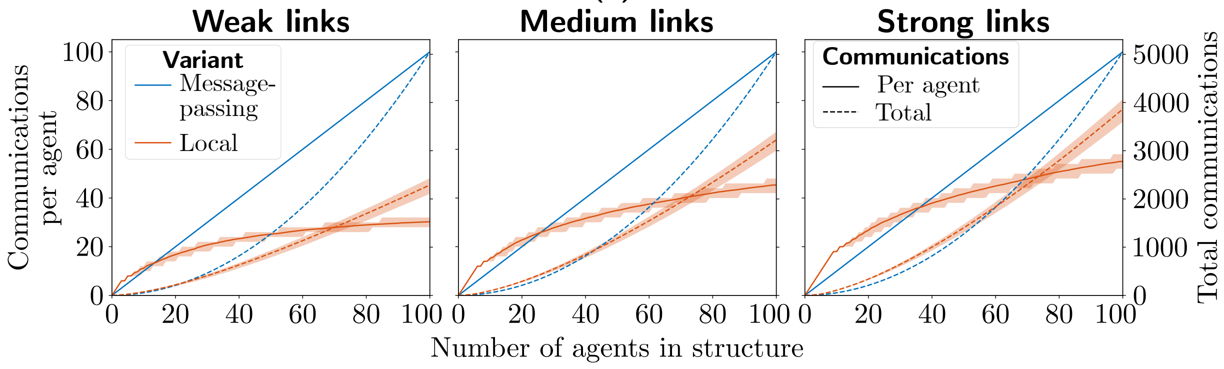
Figure 3.13. A structure of 100 agents produced by the message-passing variant of the sequential algorithm with links of medium strength. The agents are shown blue, and links are coloured by their criticalness. © 2021 IEEE.



(a)



(b)



(c)

Figure 3.14. Performance comparison between the message-passing and local variants of the sequential cantilever construction algorithm, presented for 400 trials of each variant. **(a)** Lengths of the longest stable structure as more agents are added, compared to the optimal achievable lengths:. © 2021 IEEE. **(b)** The maximum moment and axial force criticalness throughout the construction by each variant. Bars show mean values, and black lines show the 95 % confidence intervals. **(c)** The number of inter-agent communications required as more agents are added to the structure. In **(a & c)**, lines show the mean values across the trials, and the shaded regions show the 5th and 95th percentiles

of agents to self-assemble a stable structure of a given length: the average length is slightly below the optimum achievable for a given number of agents, with relatively little variation between trials. Figure 3.14b shows that the two variants exceed the allowable limits by similar amounts within each limit pair in all intermediate unstable states. The local variant hence achieves comparable performance to the message-passing variant. It can therefore be predicted that the performance of the parallel algorithm will not be hindered by only transmitting forces within links of exterior agents to active agents.

The amount of inter-agent communication when following each algorithm is shown in Figure 3.14c. The number of communications required when following the message-passing variant scales quadratically with the number of agents in the structure, as was predicted. However, the local variant does not show its theoretical worst-case performance. Instead, the number of communications for each additional agent levels off as more agents are added, meaning the total number of communications is lower than in the message-passing variant. This effect is more apparent for longer structures and those with weaker links, as they require more reinforcement and thus exhibit a greater difference between n and L_n. This will have the effect of reducing the energy usage of the system, and also decreasing the likelihood of communication errors causing problems with self-assembly. In addition, the force-aware self-assembly algorithm of Inou *et al.* also required $\mathcal{O}(N^2)$ communications, so the local variant of the sequential algorithm shows advantages over this approach [23].

Another interesting consideration is how effective the method of calculating the probability mass function $p_{column}(c)$ is. To investigate, a modified version of the message-passing variant was also implemented where $p_{column}(c)$ is calculated in a much simpler manner. In this case, a single simplified combined urgency distribution $\hat{\boldsymbol{\nu}}^*$ is calculated as:

$$\hat{\boldsymbol{\nu}}^*|_c = \max_{\substack{\alpha\in\{M,F\},\\ \beta\in\{row,column\}}} \left(\gamma_c^{\alpha,\beta}\right) \tag{3.11}$$

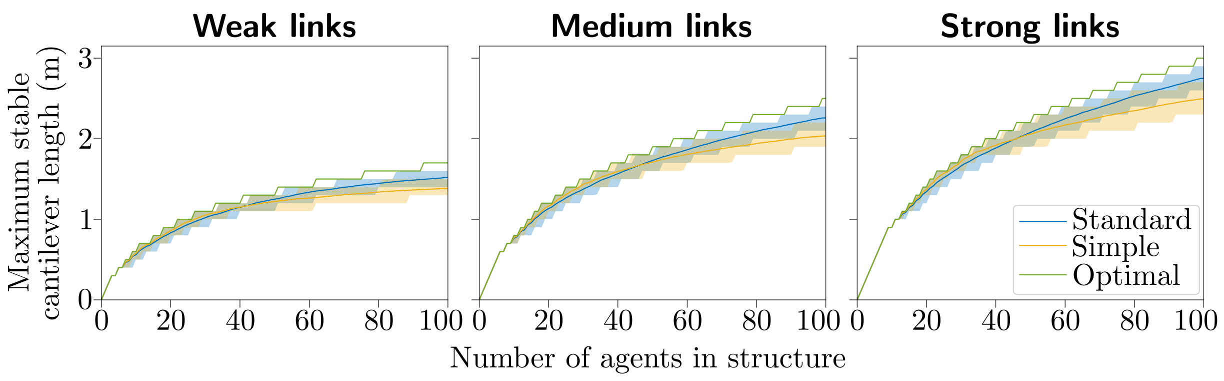
The probability mass function $p_{column}(c)$ is calculated by dividing each component of $\hat{\boldsymbol{\nu}}^*$ by the sum of all the components.

The performance of this simple method of calculating $p_{column}(c)$ compared to the standard method is shown in Figure 3.15: it shows the simple method builds slightly faster initially, but the standard method quickly reaches greater lengths for the same number of agents as more are added. It can also be seen that the maximum γ throughout the construction process is higher when the simple method is used, meaning the structures must be built with more conservative estimates of $M_{allowable}$ and $F_{allowable}$ to avoid collapse. The slight increase in computation required to implement the standard method is therefore deemed worthwhile, and carried forward to the parallel algorithm.

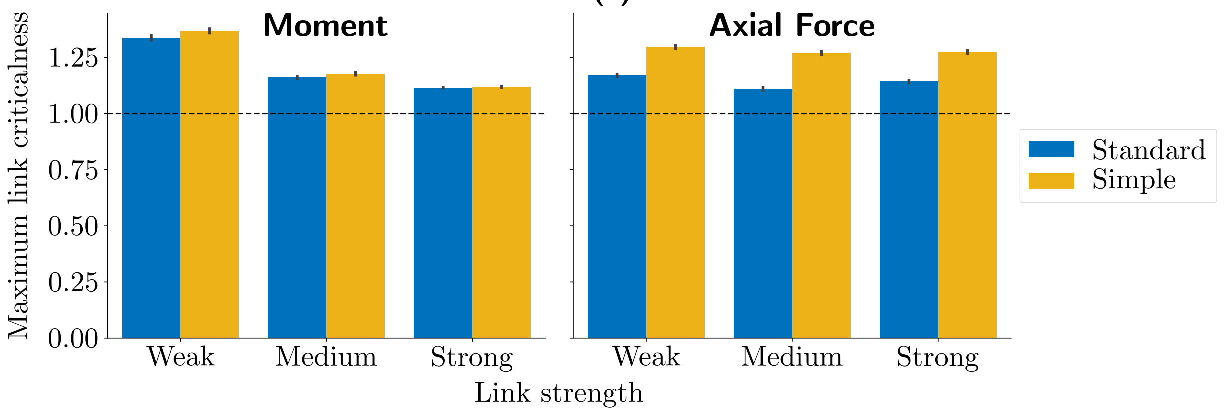
3.5.2 Parallel Algorithm Performance

In the previous section, it was demonstrated that the sequential algorithm can construct stable cantilevers of near optimum length, and the idea that agents should place near columns of high γ was validated. The parallel algorithm extends these concepts.

Frames from an example trial of the parallel algorithm are shown in Figure 3.16. The trial begins by constructing a cantilever only one agent thick, before reinforcement is required in row 2. Timesteps 62 – 64 illustrate how two agents travelling in opposite



(a)



(b)

Figure 3.15. Performance comparison between the message-passing variant of the sequential cantilever construction algorithm with $p_{column}(c)$ calculated using the standard and or simple methods, presented for 400 trials of each method. **(a)** Lengths of the longest stable structure as more agents are added, compared to the optimal achievable lengths: lines show the mean value across the trials, and the shaded regions show the 5th and 95th percentiles. **(b)** The maximum moment and axial force criticalness throughout the construction by each method. Bars show mean values, and black lines show the 95 % confidence intervals.

directions bypass each other by swapping information. The `placing` agent has calculated it can place at the tip to extend the structure, while the `gathering` agent is trying to reach column 1. The agents are therefore travelling in opposite directions, so spend timestep 63 in the `swapping` mode, were they exchange information with each other. In the following timestep, they are able to continue. The algorithm continues in this manner, correcting for any unstable configurations that occur by choosing columns to reinforce based on the force information received. The swapping procedure is able to allow agents to pass each other and continue with construction, even when active agent congestion occurs. The final agent places itself in timestep 852, causing the trial to finish.

Figure 3.17 shows a comparison of the sequential and parallel algorithms in which 100 trials of up to $N = 100$ agents were performed for each of $\delta \in \{4, 6, 8, 10, 12, 14\}$ timesteps. These are compared to the local variant of the sequential algorithm, with two modifications to allow for a fairer comparison:

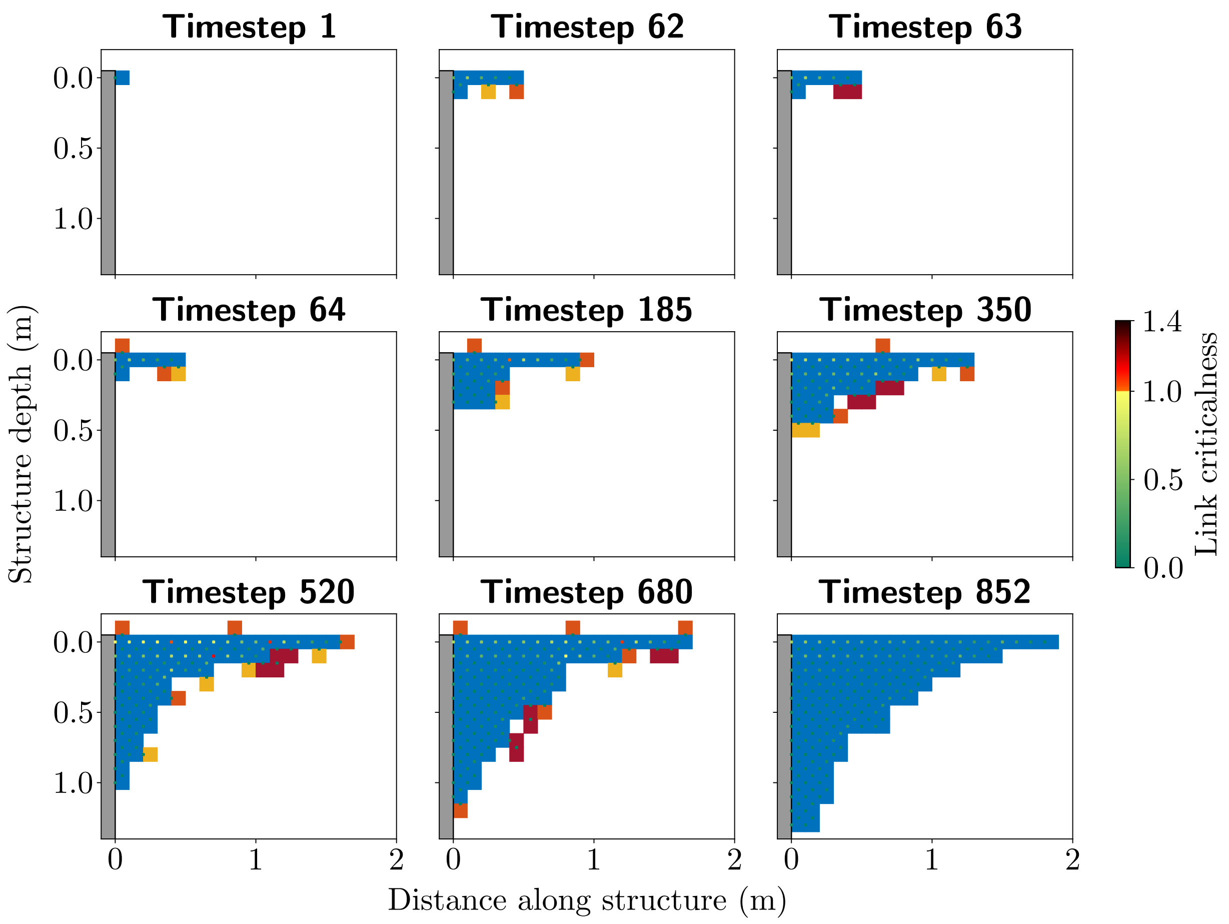


Figure 3.16. Frames from a cantilever construction sequence produced by the parallel algorithm for links of medium strength, $\delta = 8$ timesteps. Placed agents are shown blue, and active agents are coloured by their mode: `gathering` (orange), `placing` (yellow), and `swapping` (red). The fixed support is grey, and links are coloured by their criticalness.

1. Sensors are not sampled and held until the active agent passes over them. Instead, the active agent has an active link as in the parallel algorithm, and its own weight is included in the measurements.

2. Each active agent remembers which locations are valid for it to place in as it moves over the structure. When $p_{column}(c)$ is calculated, it is sampled from repeatedly until a valid column is found before the active agent starts to move. Consequently, the active agent can travel directly there.

Figure 3.17a shows that the final length of the structure increases as δ increases, and is greatest in the sequential case. However, this slight decrease in cantilever length is compensated for by the significantly lower construction time achieved by the parallel algorithm (Figure 3.17b). Stronger links tend to require a larger δ before they can achieve a comparable length to the sequential algorithm. This is partly because agents are more likely to timeout and place where they are due to congestion for small δ: as shown in Figure 3.17c, when $\delta = 4$ timesteps, up to around 70 % of agents place in this manner, but this drops to near zero for $\delta \geq 10$ timesteps, regardless of link strength. The increased frequency of agent timeout means agents place before they reach their intended placement

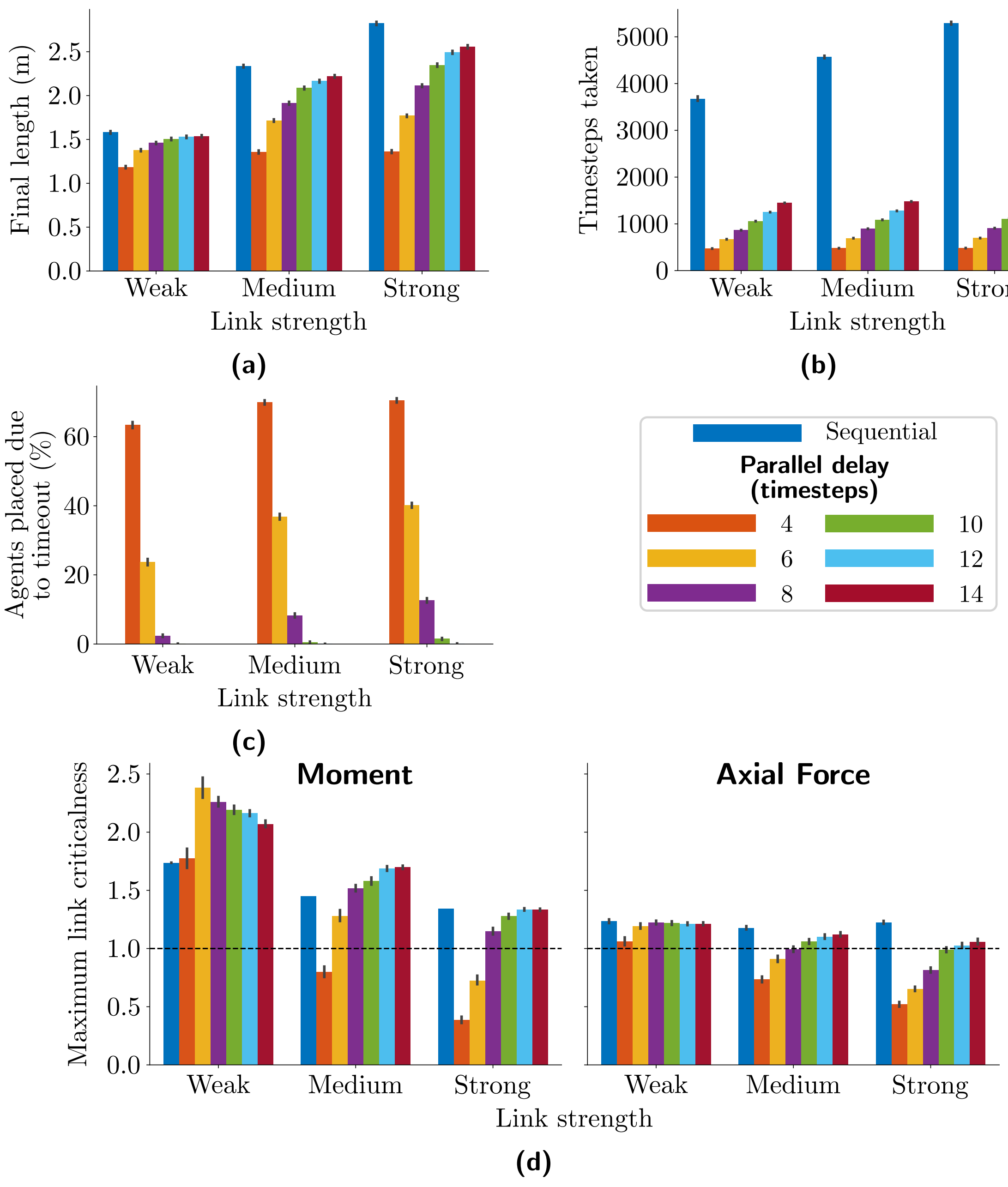


Figure 3.17. Performance comparison between the local variant of the sequential cantilever construction algorithm and the parallel cantilever construction algorithm with varying delay between when agents are added, presented for 100 trials of each setup. **(a)** The final length of structure achieved, **(b)** the number of timesteps taken for all agents to place themselves, **(c)** the percentage of agents that placed themselves due to timeout, and **(d)** the maximum moment and axial force criticalness throughout the construction. Bars show mean values, and black lines show the 95 % confidence intervals. **(a, b, & d)** © 2021 IEEE.

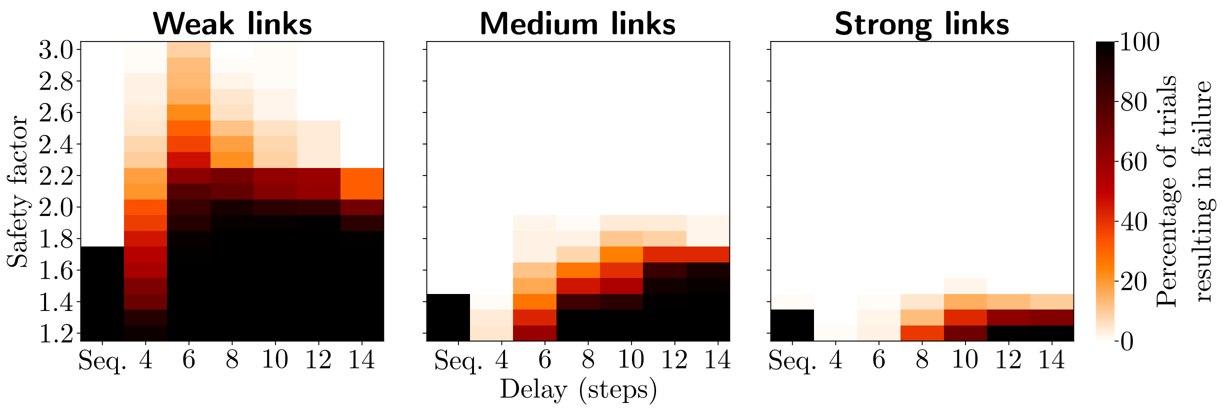


Figure 3.18. The percentage of trials in which the structure would have failed for a range of safety factors. As in Figure 3.17, results are shown for both the sequential algorithm (labelled *Seq.*) and the parallel algorithm with the given δ.

location. This leads to short and tall structures, which as shown in Section 3.3.3 are good for weak links, but creates less efficient structures for stronger links.

Another consideration is the maximum γ in the structure as it is assembled (Figure 3.17d). For weak links, the parallel algorithm creates structures that exceed $M_{allowable}$ and $F_{allowable}$ by a greater amount during construction than in the sequential case, but this exception decreases as δ increases. For strong and medium links, the maximum γ during the construction sequence is usually less than in the parallel case, and increases with larger δ. This is another result of the increased rate of agent timeout at low δ creating short and tall structures. For all link strengths, the maximum γ during construction tends to a value close to that of the sequential algorithm as δ increases.

Recall that the allowable limits are set below the actual failure strength of the links: Figure 3.18 explores how much lower these thresholds should be set. A metric called the *safety factor* is defined as the factor by which the allowable limit can be exceeded before links will actually break. It can be seen that a higher safety factor, stronger links, and lower δ generally results in fewer trials experiencing failure. This reflects the general trends observed in Figure 3.17d, and is again due to the high rates of agent timeout creating structures that are short but tall. While no trials with medium and strong links failed for safety factors greater than 2, higher safety factors were required to ensure this with weak links: typically a safety factor of 3 is sufficient, but for $\delta = 6$ timesteps a safety factor of 4 is required, as one trial experienced a maximum γ^M of 3.95. There was less variation between trials in the sequential algorithm, so each given safety factor resulted in failure in either all or no trials.

3.6 Summary

This chapter has demonstrated how cantilevers can be self-assembled by force-aware robotic agents. Two novel distributed self-assembly algorithms were described, which consider local force information to ensure links between agents do not break. In the sequential algorithm, only one agent could move at once. Two variants of this algorithm were developed, with the local variant requiring agents to coordinate and communicate to a lesser degree than the message-passing variant. The parallel algorithm was built upon the local variant of the sequential algorithm, and allowed for multiple agents to move simultaneously. A custom simulator was developed to evaluate these algorithms, which modelled the structure as a truss to efficiently calculate the moment and axial force in links between agents. Optimal cantilevers were also computed to compare the performance of the self-assembly algorithms against.

It was found that both algorithms were able to induce self-assembly into stable cantilevers, the length of which was shown to be near the optimal length for a given number of agents. Both variants of the sequential algorithm displayed similar trends, despite the reduced coordination and communication required. The parallel algorithm was shown to result in cantilevers of comparable length to the sequential case, but considerably quicker. The maximum moment and axial force during construction following the parallel algorithm was similar to that occurring during execution of the sequential algorithm.

The self-assembly algorithms presented in this chapter will be verified in real-life in Chapter 6. Prior to this, the next chapter considers how the structure should evolve once the other side of the chasm is reached. One can predict that a bridge supported at both ends will require fewer agents to maintain stability when compared to a cantilever of equivalent length. The safe deconstruction of the bridge when it is no longer required will also be considered.

Chapter 4

Bridge Optimisation

The previous chapter presented distributed algorithms that enable a group of self-assembling modular robots to build a cantilever that spans across a void (Figure 4.1a). These algorithms incorporate local force measurements to ensure the structure does not collapse under its own self-weight. This chapter builds on this work by considering how the structure can be adapted when it reaches the other side of the void to form a bridge. Once the structure becomes supported at both ends, it is likely to benefit from *bridge optimisation*. This is the name given to the process where agents are removed or repositioned to produce a stable structure containing fewer agents than when the cantilever initially reached the other side of the gap (Figure 4.1b). This frees agents to complete other tasks on the opposite side of the gap, while ensuring the remaining structure is strong enough to maintain stability as agents cross it.

The effectiveness of this procedure stems from the additional reinforcement offered by the support on the other side of the gap from which the cantilever was constructed. A simplified representation of a generic bridge of length L between two supports loaded by a UDL of magnitude w per unit horizontal length is shown in Figure 4.2. The shear force and moment at the supports of this bridge, $S_{s,\vdash\!\dashv}$ and $M_{s,\vdash\!\dashv}$ respectively, can be obtained from standard results [189] as:

$$S_{s,\vdash\!\dashv} = \frac{wL}{2} = \frac{S_{s,\vdash}}{2} \tag{4.1}$$

$$M_{s,\vdash\!\dashv} = \frac{wL^2}{12} = \frac{M_{s,\vdash}}{6} \tag{4.2}$$

These values are compared to their equivalents for a cantilever $S_{s,\vdash}$ and $M_{s,\vdash}$, given in (3.1) and (3.2) respectively. This demonstrates that the extra fixed support significantly reduces the shear force and moment at each support. The internal stresses will therefore also be lower, thus the thickness of the structure at certain points can be reduced while maintaining these stresses at safe levels. This is the aim of the bridge optimisation algorithm.

This chapter is organised as follows. The problem considered in this chapter is formally defined in Section 4.1, then in Section 4.3 optimal bridges are found to compare the performance of the self-assembly algorithm to. The distributed force-aware bridge optimisation algorithm is described in Section 4.4, and verified in simulation in Section 4.5. Section 4.6 summarises the chapter.

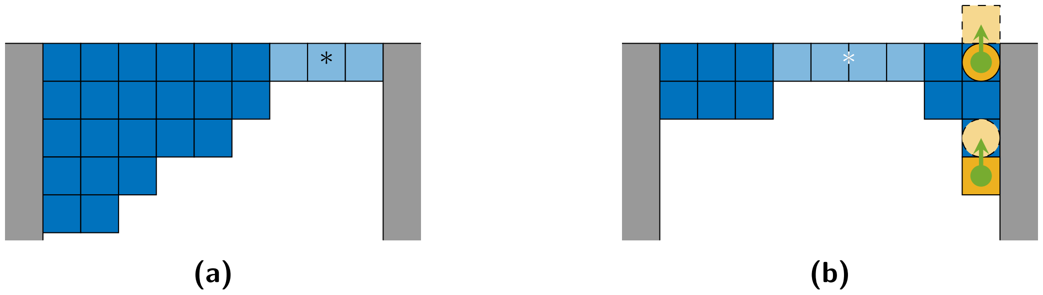


Figure 4.1. **(a)** The bridge optimisation algorithm starts with a cantilever that has just reached the other side of a void. **(b)** Agents reconfigure or remove themselves to leave a more slender bridge that will not collapse under its own weight. Agents are shown in blue and the fixed supports in grey. The lighter portion in each diagram shows the narrow section, while the inflection points are marked with asterisks. Active agents are shown yellow, and are drawn as a circle when following the paths shown in green to pass through the structure in the third dimension.

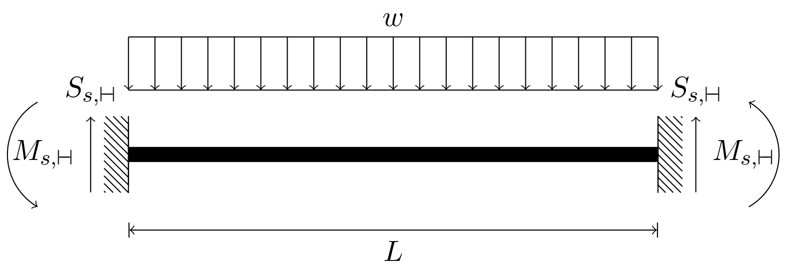


Figure 4.2. A simple bridge of length L, loaded with a UDL of magnitude w per unit horizontal length. This loading causes equal shear forces and moments at each support, denoted $S_{s,\mathsf{H}}$ and $M_{s,\mathsf{H}}$ respectively.

The work in this chapter is based on sections of the author's published work [190], with certain aspects expanded upon. In particular, this chapter draws from Sections II, III, IV.A, IV.D, IV.E, V.A, and V.C of this work.

4.1 Problem Formulation

This chapter considers a similar 2D grid of homogeneous agents to that considered in Section 3.1. There are a number of additions made to this situation as described below.

The cantilever construction algorithms began with a single agent in position $(1, 1)$ connected to a single fixed support. The starting point of bridge optimisation is an existing configuration of length L between two fixed supports, occupying positions $\{(r, c) \mid r > 0 \land c \in \{0, L + 1\}\}$.

Agents can again be placed or active. Active agents during cantilever construction initialised in position $(0, 0)$, but bridge optimisation aims to reduce the number of agents in the structure, so no more agents are added. Instead, placed agents become active when they deem it necessary. Active agents become placed when they reach a suitable location.

As for cantilever construction, structures must be continuous. However, the definition is extended to reflect the additional fixed support: each placed agent must connect to the top of its column by a vertical chain of placed agents, and to *either* fixed support by a horizontal chain of placed agents. It must also contain a single contiguous region with only one placed agent in each column, referred to as the *narrow section* (columns 7 – 9 in Figure 4.1a and columns 4 – 7 in Figure 4.1b). Either side of this, the number of placed agents in each column increases monotonically, so the centre of the narrow section is called the *inflection point* (column 8 in Figure 4.1a and between columns 5 and 6 in Figure 4.1b). These restrictions are imposed so that the structures resemble arches, a strong shape commonly found in bridges seen in everyday life. Empty locations with a placed agent on either side are referred to as *canyons*. Active agents can move through canyons, but cannot place in them without violating the requirement for a narrow section.

During cantilever construction, active agents move using a 2D sliding square model, where in each simulation timestep they can travel around the structure by moving into an unoccupied cell in their Moore neighbourhood if in doing so they will remain adjacent to another agent. This remains possible during bridge optimisation, but agents also have an additional way of moving. Agents must be able to pass between the top and bottom perimeters of the structure so that they can leave the environment, but this cannot be done without additional consideration since the structure is supported at both ends. This could be achieved by coordinating placed agents to shift along the column in question, but this would require a high degree of inter-agent coordination, so is undesirable. Instead, it is chosen to allow active agents to move into the third dimension in front of existing placed agents (Figure 4.1b). Moving into the third dimensions in other scenarios is not permitted, as constructing 3D structures is beyond the scope of this work.

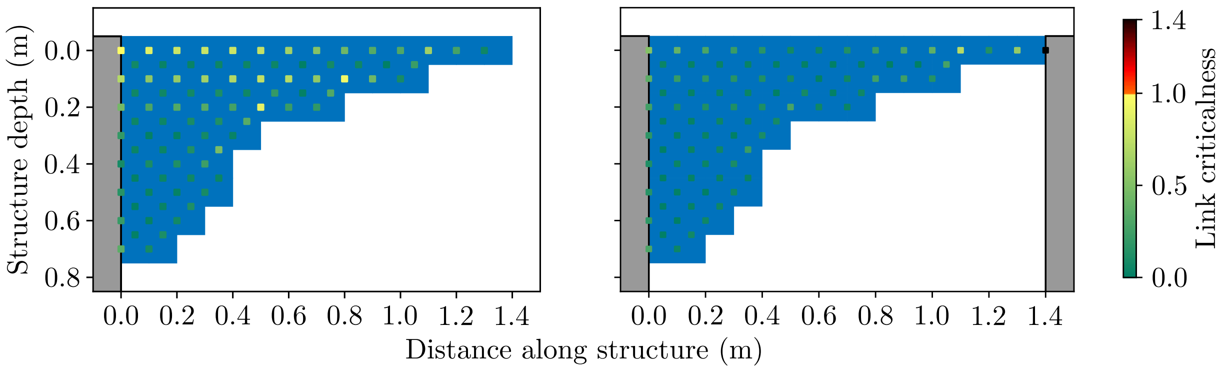
Agents are once again able to sense the moment M and axial force F in their links. These data are used to calculate a criticalness γ according to (3.4) to determine if the structure is stable or unstable. The bridge optimisation algorithm reduces the number of agents in an initial configuration spanning between the two fixed support surfaces, while aiming to maintain stability.

4.2 Simulation Environment

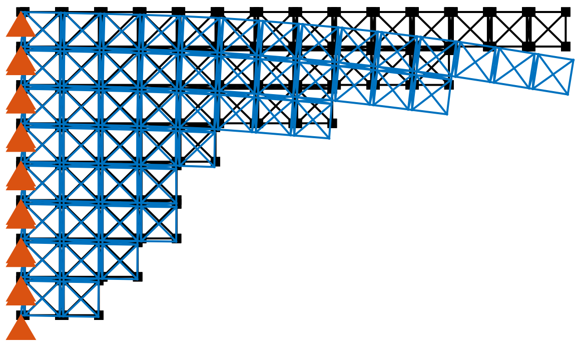
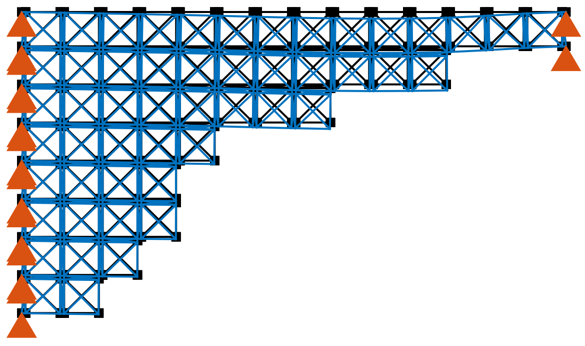
The algorithm is evaluated using the same truss-based simulator developed for the cantilever construction algorithms, described in Section 3.2 and Appendix A. The same limitations are present here, but the assumption that structures do not deform has an additional consequence for bridges. When a cantilever reaches the right support, the deflection can cause the link connecting the tip of the structure to this support to become critical immediately upon its formation. This is because in reality the cantilever will sag towards the tip. This means that when the cantilever connects to the right support, it will sometimes experience high forces within the links near this support that were previously lower, as these links sustain the forces required to lift the tip to the correct height.

An example of this phenomenon is shown in Figure 4.3. In Figure 4.3a, a structure is seen before and after connecting to the right support. Note how it is stable before (left), but the link that connects to the right support is critical (right). The reason for this is revealed by Figure 4.3b: the cantilever sags towards the tip, but when connected to the right support these nodes are raised up. This exerts additional forces on them, so the link forming the connection to the support becomes critical.

This phenomena is acceptable for the simulated structures, but is highlighted here



(a)

(b)

Figure 4.3. An example illustrating how the same structure can be stable as a cantilever, but unstable when the connected to the right support. **(a)** A structure before and after connecting to the right support. Agents are shown blue, the fixed supports grey, and links are coloured by their criticalness assuming weak links. **(b)** the truss approximation of the corresponding structures, shown both undeformed (black) and deformed (blue), with the deformation exaggerated for clarity. The attachments to the fixed supports are shown in orange.

as it may affect the design of physical robot systems that employ this algorithm. Such robotic agents should be able to automatically correct for small alignment errors caused by this sagging when connecting to the right support.

4.3 Offline Structural Optimisation

Optimal bridges were generated offline as a baseline to compare the performance of the distributed bridge optimisation algorithm against. The optimal configurations of length L are defined to be those that are stable, continuous as defined in Section 4.1, and consist of the fewest number of agents N.

4.3.1 Optimisation Procedure

Optimal configurations are generated by exhaustive search, similar to how optimal cantilevers were found in Section 3.3. The process begins by selecting an L, $M_{allowable}$, and $F_{allowable}$, then choosing an N that is believed might be able to build a stable cantilever

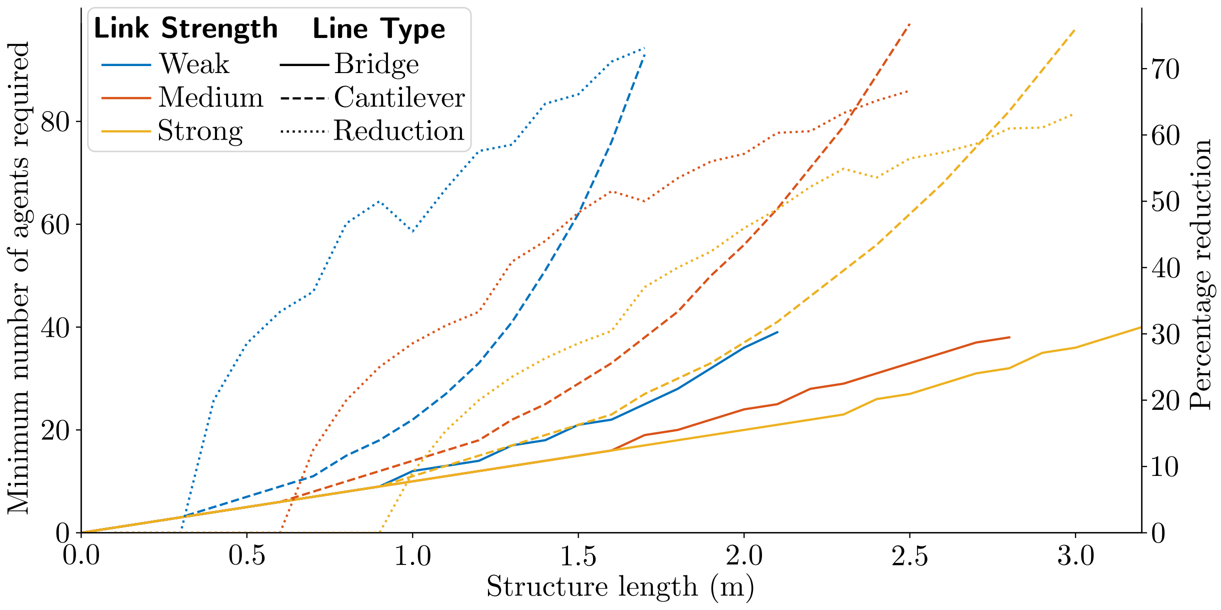


Figure 4.4. The minimum number of agents required to build stable bridges (solid lines) or cantilevers (dashed lines) of a given length. Dotted lines show the percentage of agents in an optimal cantilever that can be removed to leave an optimal bridge of the same length.

of this length and link strength. Configurations are generated by setting the number of agents in row 1 as L, and specifying the number of agents in the lower rows connected to the left and rights supports N_L and N_R respectively. All configurations of N_L agents are enumerated using integer partitioning [187] by iterating over $\lceil \frac{N-L}{2} \rceil \leq N_L \leq (N-L)$: the i^{th} component of each partition represents the number of agents in row $i+1$ connected to the left support $N_{L,i}\ \forall\ 1 \leq i \leq N_L$. For each of these configurations, all configurations of $N_R = N - L - N_L$ are all enumerated, giving the number of agents in row $i+1$ connected to the right support $N_{R,i}$. These partitions are arranged such that $N_{L,i} \leq N_{L,i+1}$ and $N_{R,i} \leq N_{R,i+1}$ to satisfy the continuity condition. Configurations in which $N_{L,i} + N_{R,i} > L$ are discarded as they are non-physical. It is possible that $N_{L,i} + N_{R,i} = L$, so the optimisation procedure does not assume that the structure contains a narrow section.

Due to symmetry, these iteration ranges are sufficient to enumerate all possible configurations. Each configuration is analysed to find the maximum γ in all its links. If a stable configuration is found then L is incremented, otherwise N is incremented. Initially, it is chosen to set $L = N = 1$ to find the minimum N required to build stable structures for all $1 \leq L \leq L_{max}$ for this link strength up to a specified L_{max}.

4.3.2 Resulting Optimal Bridges

Optimal bridges were generated for weak, medium, and strong links as defined in Section 3.3.2 for $N \leq 40$ agents. As shown in Figure 4.4, this allows for optimal bridges to be compared to all lengths of optimal cantilever generated in Section 3.3, which are those with $N \leq 100$ agents. As expected, stronger links allow bridges of a given L to be built with fewer agents. Additionally, fewer agents are required to build stable bridges than cantilevers of equal L and link strength, with the difference increasing for longer structures. For the structures considered here, up to 73 % of the agents in an optimal cantilever can be removed to create an optimal bridge of equal L.

Figure 4.5 shows a selection of optimal bridges, compared to the optimal cantilever of the same length and link strength; a complete list of optimal bridges is given in Appendix B. As expected, the optimal bridges are more slender than the equivalent optimal cantilevers, containing significantly fewer agents. For all optimal bridges found, not just the bridges shown in this figure, only the first row spans the whole gap. This implies that additional full rows are inefficient, at least for the range of L and link strengths considered here. This supports the constraint that all valid bridges produced by the bridge optimisation algorithm include a narrow section.

4.4 Algorithm Design

This section describes the bridge optimisation algorithm. Firstly, the result derived using structural mechanics in Section 3.4.1 is extended to consider the case when a structure is supported from both ends. After this, the algorithm is presented in detail.

4.4.1 Theoretical Basis

In Section 3.4.1, a cantilever with a height that continuously and monotonically decreased towards the tip that is loaded by a UDL was analysed. It was shown that increasing the height at a particular location decreases the longitudinal stress at that point. This inspired the design of the cantilever construction algorithms to place agents at the bottom of columns near to locations where the links were critical. This result is now extended to consider structures supported at both ends.

Theorem 2. *Consider a bridge of length L between two vertical fixed supports. The height of the bridge $h(x)$ a distance x from the left support is continuous, and has a single inflection point, as shown in Figure 4.6. The bridge has constant breadth b into the page, and the only load is a UDL of magnitude w per unit horizontal length. For such a bridge, increasing the height $h(x)$ at a distance $x = x_0$ from the left support causes a decrease in the maximum longitudinal stress experienced in the cross-section here, $\sigma_{max}(x_0)$.*

Proof. As for the cantilever in Section 3.4.1, this bridge can be analysed using elastic beam theory [188]. The first step is to again make a cut through the cross-section at x, but in contrast to the analysis of the cantilever, the portion left of the cut is retained (Figure 4.6b). In order to calculate the internal moment $\tilde{M}(x)$, the shear force S_s and moment M_s at the support must be known. Fortunately, the loading configuration is a standard case so these can both be obtained from [189] as:

$$S_s = \frac{wL}{2} \tag{4.3}$$

$$M_s = \frac{wL^2}{12} \tag{4.4}$$

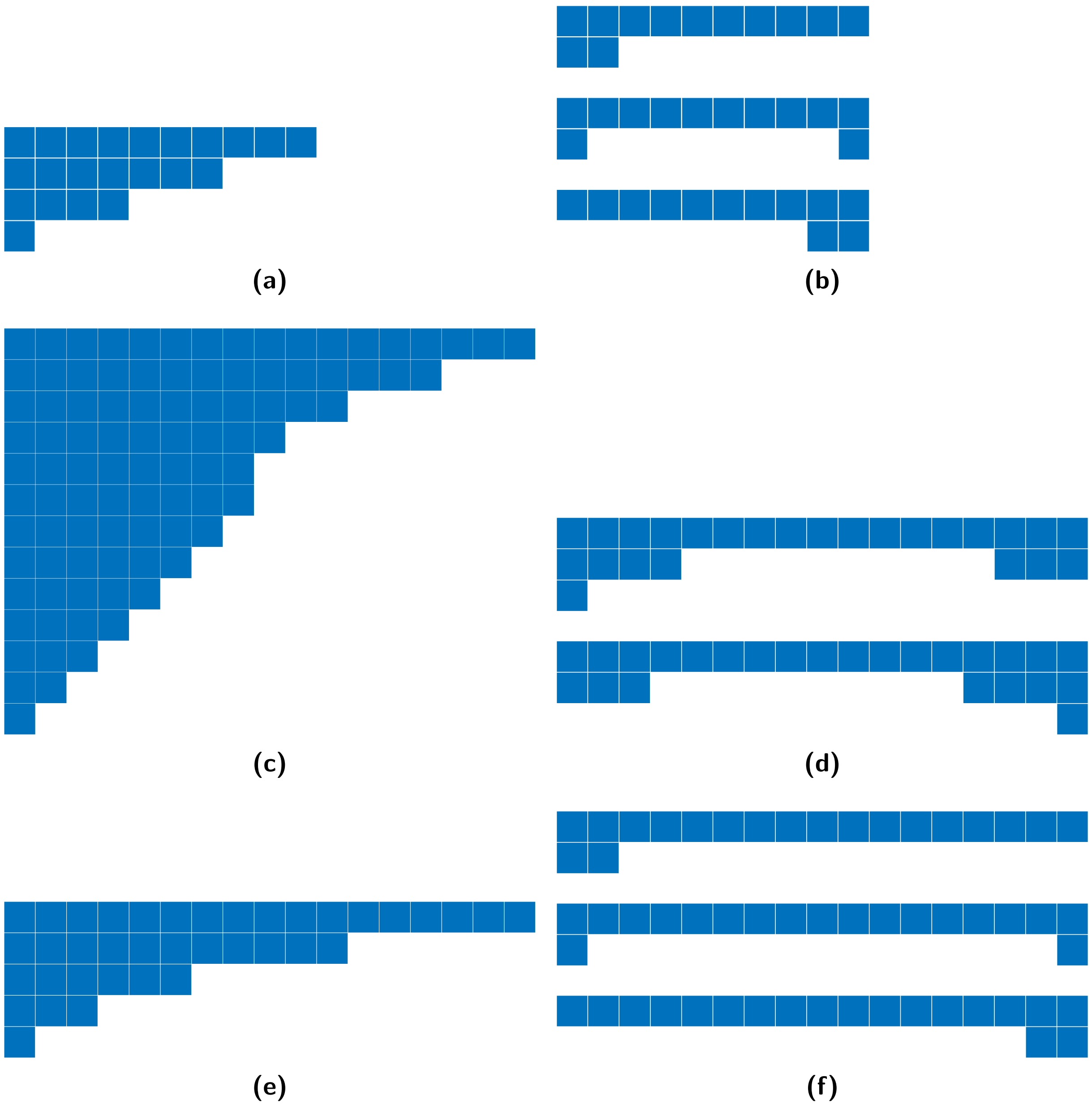


Figure 4.5. Optimal structures with a given number of agents for different limit pairs. **(a)** An optimal cantilever with weak links requires 22 agents to reach a length of 1 m, whereas a similar optimal bridge **(b)** requires only 12 agents in three possible configurations. **(c)** Increasing the length to 1.7 m requires 93 agents in a cantilever, while **(d)** either of the two optimal bridges can be built with only 25 agents. **(e)** An optimal cantilever of this length constructed with links of medium strength requires 38 agents, whereas three optimal bridges of this length and link strength can be built with only 19 agents.

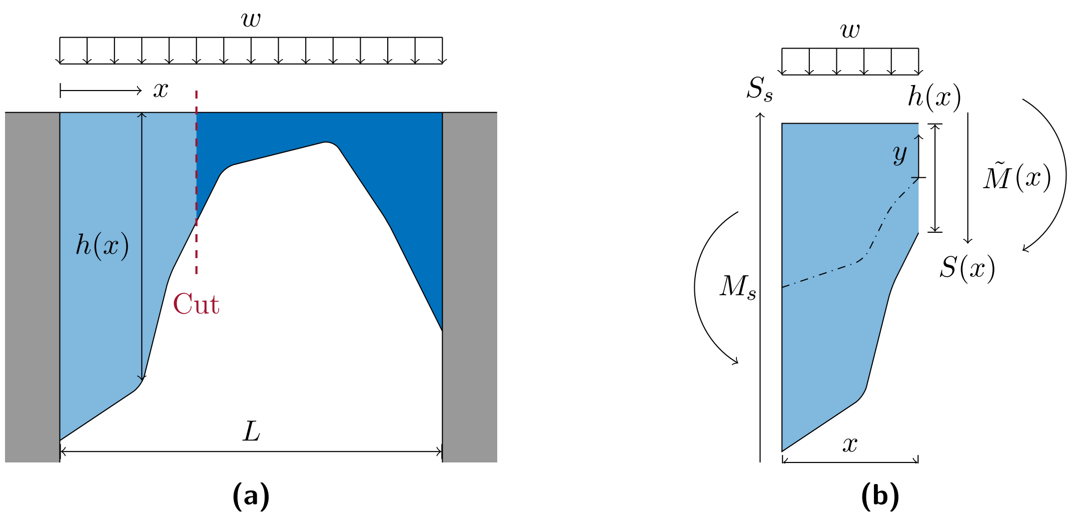


Figure 4.6. (a) Profile of a bridge (blue) with continually-varying height $h(x)$ a distance x from the left end extending from a fixed support (grey). The only load is a UDL of magnitude w per unit horizontal length. **(b)** A cut through this bridge at x, showing the internal shear force $S(x)$ and moment $\tilde{M}(x)$ on the exposed cross-section, as well as the shear force and moment at the support (S_s and M_s respectively). The dashed line shows the neutral axis, which is assumed to be in the centroid of the cross-section.

It is now possible to calculate $\tilde{M}(x)$ through static equilibrium:

$$\tilde{M}(x) + S_s \cdot x = M_s + wx \cdot \frac{x}{2}$$
$$\tilde{M}(x) = \frac{w}{12}\left(6x^2 - 6xL + L^2\right) \tag{4.5}$$

As in the analysis for cantilevers, the neutral axis is assumed to be at the centroid of the cross-section. At each x, $\sigma_{max}(x)$ occurs at the greatest possible distance from the neutral axis y. This value can therefore be calculated using the second moment of area of the cross-section I as:

$$\sigma_{max}(x) = \frac{\tilde{M}(x)y}{I(x)}$$
$$\sigma_{max}(x) = \frac{\frac{w}{12}\left(6x^2 - 6xL + L^2\right) \cdot \frac{h(x)}{2}}{\frac{bh(x)^3}{12}}$$
$$\sigma_{max}(x) = \frac{w\left(6x^2 - 6xL + L^2\right)}{2bh(x)^2} \tag{4.6}$$

This shows that, just as for cantilevers, $\sigma_{max}(x_0)$ decreases as $h(x_0)$ increases in bridges supported at both ends. □

The same caveat that applied to the cantilever analysis also applies here: the equations are derived for an idealised scenario with a bridge of continuously-varying height under a UDL. However, this result demonstrates that placing agents at the bottom of columns where links are critical is expected to reinforce them. For bridge optimisation, it is useful to note that the opposite is also true: decreasing $h(x_0)$ will increase $\sigma_{max}(x_0)$. This implies that removing agents from the bottom of a column will likely bring the links in that column closer to becoming critical. Agents should therefore only reposition themselves if the links of the column they are in are not close to becoming critical.

4.4.2 Algorithm Description

Two cantilever construction algorithms were designed: the sequential and parallel algorithms. Implementing a sequential bridge optimisation algorithm would require significant coordination and a large number of messages to be passed between placed agents to ensure only one agent becomes active at a time, so it was chosen to only develop a parallel algorithm. As in the parallel cantilever construction algorithm, active agents *advance* each timestep in a randomised order, and transition through several *modes* as described below. While moving, they track their location and the heights of columns that they visit so as to determine which side of the inflection point they are on. At the end of each timestep, M and F in links is updated for all agents.

Placed agents on the lower perimeter with an empty cell on their left or right can become active if they believe they are not significantly contributing to the strength of the structure: they are therefore called *release candidates*. Every timestep, each release candidate j will release itself from the structure with probability $P_{release,j}$. This is calculated from the maximum γ across each link of agent j, $\gamma_{j,max}$, as:

$$P_{release,j}(\gamma_{j,max}) = \begin{cases} \mu e^{-\epsilon \gamma_{j,max}} & \text{for } \gamma_{j,max} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

This function has two shape parameters: ϵ affects the rate the function decays with $\gamma_{j,max}$, and $\mu = P_{release,j}(0)$. Throughout this chapter, it is chosen to set $\epsilon = 10$ but to explore the effect of changing μ to examine how the number of simultaneous active agents in the simulation affects performance.

Each release candidate that successfully releases itself becomes active and disconnects all but one of its links. As shown in Figure 3.9, this means it is no longer offering any support to the structure. This remaining link becomes its *active link*, the choice of which is explained more in Section 4.4.2.A. In the next timestep, the newly-active agent begins to follow Algorithm 4.1.

Active agents start in the `releasing` mode (Lines 2 – 9). They first check the readings of M and F in links of the agent above. If the release has caused any of these links to become critical, they immediately place back where they are. Otherwise, they swap to the `gathering` mode and take a step for the first time.

Agents that are `gathering` move around the lower perimeter of the structure and communicate with the placed agents above them to obtain the M and F values recorded by their sensors (Line 11). These are stored in four arrays $\{\boldsymbol{\gamma}^{\alpha,\beta} \ \forall \ \alpha \in \{M, F\} \wedge \beta \in \{\mathit{row}, \mathit{column}\}\}$, as was the case for cantilever construction. Agents travel left until they reach column 1, then move right to column L to obtain information about the whole structure (Line 20). When agents pass over columns they have already visited, the information in $\boldsymbol{\gamma}^{\alpha,\beta}$ for this column is updated to the most recent values.

When an active agent reaches column L, it decides what to do based on $\boldsymbol{\gamma}^{\alpha,\beta}$. The bridge optimisation algorithm aims to remove agents while retaining structural stability. Active agents therefore swap to the `escaping` mode if no links were measured to be critical

Algorithm 4.1. The bridge optimisation algorithm, based on the parallel cantilever construction algorithm (Algorithm 3.3)

```
switch mode do
    case releasing do
        Record $M$ and $F$ from placed agent above into $\gamma_c^{\alpha,\beta}$;
        if Agent above has a critical link then
            Place back here (agent no longer active);
            return
        else
            mode ← gathering;
            Make step*;
    case gathering do
        Record $M$ and $F$ from placed agent above into $\gamma_c^{\alpha,\beta}$;
        if Agent in column $L$ and Previously visited column 1 then
            if No measured links critical then
                mode ← escaping;
            else
                Calculate $p_{column}(c)$ from $\boldsymbol{\gamma}^{\alpha,\beta}$;
                $c_{target}$ ← sample from $p_{column}(c)$ without replacement;
                mode ← placing;
        else
            Make step*;
    case escaping do
        if In position $(0, L)$ then
            Agent leaves simulation;
            return
        Make step*;
    case placing do
        if In column $c_{target}$ then
            Attempt to place;
            if Placement succeeded then
                Agent no longer active;
                return
            else if Agent at top of canyon then
                mode ← escaping;
            else
                $c_{target}$ ← sample from $p_{column}(c)$ without replacement;
        Make step*;
    case swapping do
        mode ← previous mode;
if Agent stationary for $> \tau$ timesteps then
    Attempt placement;
    if Placement succeeded then
        Agent no longer active;
```

* Steps made if the agent is not the sole support of another

(Line 14). In subsequent timesteps, they pass through the structure here to reach row 0, then move to the right support, whereupon they exit the simulation (Lines 21 – 25).

If an active agent believes the structure is unstable when it reaches column L, it will attempt to reinforce it. This process begins by calculating the probability mass function $p_{column}(c)$ as described in Section 3.4.2.A that represents the probability of placing in column c based on the received force information: note that active agents calculate this from measurements of forces from placed agents only on the bottom of the structure, without considering the values measured by agents on the top of the structure, as was the case for cantilever construction. This function has a high probability of placement in columns where high M and F values were recorded and thus should be reinforced. Only links on the bottom of the structure are considered, but this is sufficient to make informed placement decisions, as seen when comparing the message-passing and local variants of the sequential cantilever construction algorithm (Section 3.5.1). The active agent samples from this distribution without replacement to choose a column to visit c_{target}, and switches to the `placing` mode (Lines 16 – 18).

The `placing` mode describes how agents reinforce the structure. Each active agent in this mode first checks if it has reached c_{target} and is directly below a placed agent. If so, it attempts to place here (Line 28), which will have one of three outcomes:

1. Placing here will not violate the continuity condition or requirement for a narrow section, therefore it does so (Line 30).

2. The placement location is at the top of a canyon, so placing here would violate the requirement for a narrow section. Instead, the active agent swaps to the `escaping` mode and will pass through the structure here in future timesteps (Line 33). This decreases the total number of active agents quickly when there is too much traffic to make informed decisions about placement locations. It specifically reduces congestion around canyons, which otherwise can become significant bottlenecks.

3. Placing in this column violates the continuity condition. In this situation, the agent will select another c_{target} from $p_{column}(c)$ without replacement (Line 35).

If the agent is subsequently still active, it will make a step either towards escaping the structure, or towards the newly-selected c_{target} (Line 36).

When two agents attempt to move in opposite directions past one another, they instead exchange information in order to 'become' the other agent. They enter the `swapping` mode for one timestep in which they remain stationary, modelling the real-life time cost of this communication (Lines 37 – 38). This is explained in more detail in Section 4.4.2.B.

If an agent is stationary for too long, it is deemed to have got stuck due to high numbers of nearby active agents desiring to travel in different directions. It attempts to place itself where it is to reduce this congestion (Lines 39 – 42). The number of timesteps before timeout τ is set to 20 in this chapter. If the agent is at the top of a canyon when timeout occurs, attempting to place will cause it to swap to the `escaping` mode to reduce congestion as described above.

4.4.2.A Active Links

In order to prevent active agents providing support to the structure, they choose a single face from which to attach to adjacent agents using their active link. If an agent

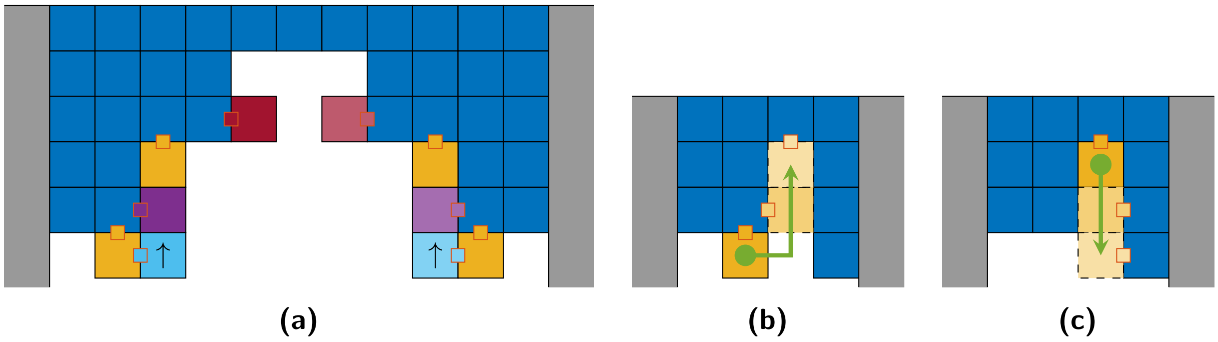


Figure 4.7. Setting active links in special circumstances. Placed agents are shown in blue, fixed supports in grey, and active agents in other colours. The active link of each agent is shown by the small orange-bordered square of matching colour. **(a)** The situations considered in cantilever construction (Figure 3.10), and their equivalents shown in paler colours when the same situation occurs on the other side of the inflection point. **(b & c)** An active agent entering then leaving a canyon, where the colour gets paler to denote the position in subsequent timesteps as the green arrows denoting the motion are followed.

is connected to the active link of another agent then it does not move so as to prevent the other becoming unsupported. Usually, the active link is set to the bottom face of the agent when it is above row 1, and on the top face otherwise. Exceptions are made when there is no agent on the usual connection face, or if connecting on this face would prevent movement in the next timestep, as shown in Figure 4.7 and described below.

- **Escaping the structure:** When agents are escaping the structure, they pass through the dimension perpendicular to the 2D plane containing the majority of the agents. The active link is therefore set to the face in the direction of this plane.

- **Mirroring of cantilever cases:** During cantilever construction, when the active agent is below row 1 its active link is set to the left face in the following situations:

  (i) There is no placed agent above.

  (ii) There is an active agent above and a placed agent on the left.

  (iii) The agent in question is moving upwards, there is an active agent on its left, and there is an active agent above it that is not swapping.

  During bridge optimisation, the mirror of these conditions about the vertical axis is also included. Active agents therefore set their active link to the right face when these situations occur right of the inflection point (Figure 4.7a, red, purple, and cyan agents respectively).

- **Canyons:** When the active agent is moving past a canyon, it moves up to row 2 then back down to the exit. Agents swap what side of the inflection point they believe they are on when they reach the top of the canyon. The active link is set to the top face when at the top of the canyon. Otherwise it is set to the left face when on the left side of the inflection point and vice versa, as would be the case if the agent was not in a canyon (Figures 4.7b & 4.7c).

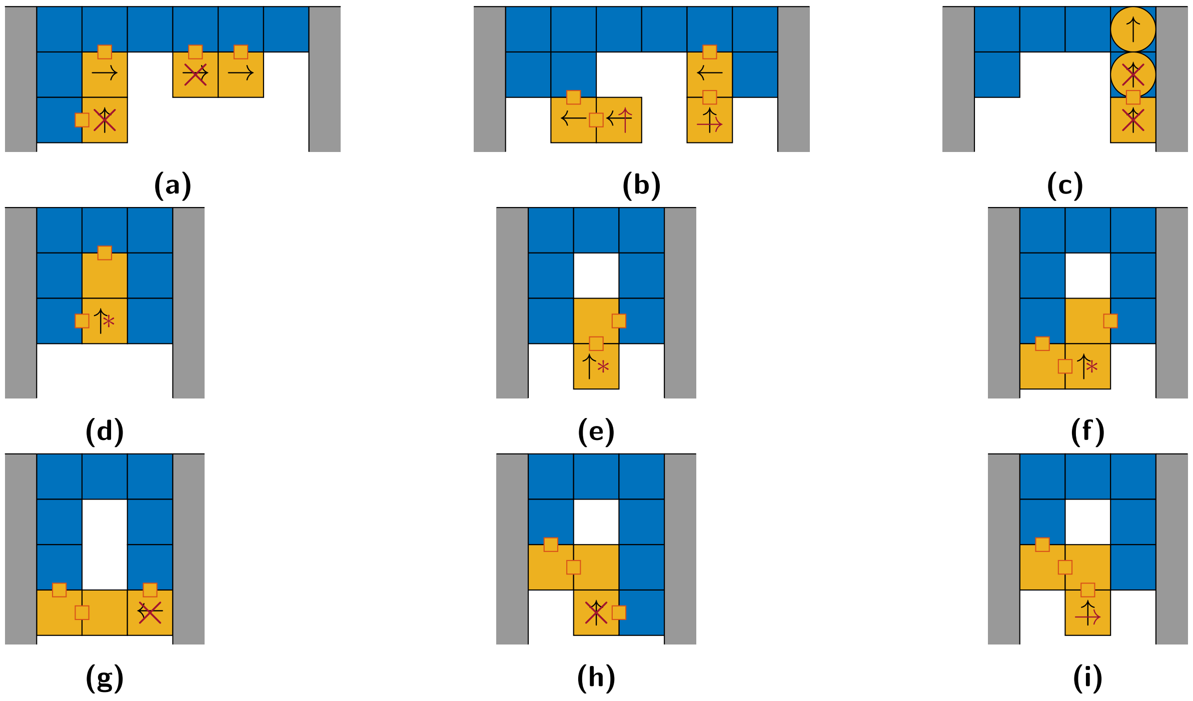


Figure 4.8. Special behaviours when agents should not swap information with each other. Active agents are shown in yellow (as a circle if they are passing through the structure), placed agents in blue, and the fixed supports in grey. Active links are shown as small yellow squares with orange borders. Salient directions of travel are shown by black arrows, and overridden directions are shown in red. Agents marked with a red asterisk will skip over the canyon instead of entering or moving further towards the top of it. Equivalent behaviours exist for the same situations mirrored about the vertical axis.

4.4.2.B Swapping

If two agents travelling in opposite directions encounter each other, they usually exchange information and 'become' one another. However, there are certain situations where they should not swap and instead remain stationary or temporarily turn around to allow space to be made for another active agent to move into. These situations are shown in Figure 4.8 and described below.

- **Direction:** Since agents advance in a random order, it is possible that two agents moving in the same direction will attempt to swap. In this situation the swap should not take place. The initiating agent will normally abort the swap and remain stationary (Figure 4.8a), but if it is attached to the agent it is trying to swap with, it will step backwards in the next timestep to allow the other agent to vacate this space (Figure 4.8b). This case also exists for cantilever construction (Section 3.4.3.B).

- **Passing through:** If an agent is blocked from passing through the structure by another active agent, it will wait for this space to clear instead of swapping (Figure 4.8c). Agents can only pass through the structure in one direction (upwards), so this is effectively a special case of the previous situation.

Link strength	**Length (m)**					
	1.2	**1.4**	**1.6**	**1.8**	**2.1**	**2.3**
Weak	✓	✓	✓	✗	✗	✗
Medium	✗	✓	✓	✓	✓	✗
Strong	✗	✓	✓	✓	✓	✓

Table 4.1. The combinations of link strength and lengths of structure the bridge optimisation algorithm was tested for.

- **Within canyons:** Canyons represent a significant bottleneck in agent motions, as each would like to pass up and down the whole canyon. In order to reduce the time that colliding agents spend in canyons, such agents do not swap here. Instead, the lower agent will pass over the canyon without visiting the top. The agents marked by an asterisk in Figures 4.8d – 4.8f are currently on the left of the canyon and would normally travel up it when they next advance. However, their path is blocked by another active agent: instead of swapping with it, they switch to the right side of the canyon and carry on with their motion. If the lower agent is in the `gathering` mode, the higher agent will also share its measurements of M and F in this column with the lower agent. If the lower agent is in the `placing` mode and has set c_{target} to this column, it instead draws another c_{target} from $p_{column}(c)$.
- **Different sides of canyon:** It is also possible that active agents could attempt to swap when they are outside a canyon but on opposite sides of it. If they are not attached to each other, the agent that is not directly below the canyon will abort the swap and remain stationary (Figures 4.8g & 4.8h). If the agents are attached, the lower one will take a step backwards (Figure 4.8i). These behaviours allow the agent directly below the canyon to vacate this space in the next timestep.

When agents step backwards, they will occasionally be required to move into a fixed support. In this case, they immediately place at this location instead, so future agents can step over them.

4.5 Simulation Results

The bridge optimisation algorithm was tested for the same weak, medium, and strong link strengths defined in Section 3.3.2. The link strengths were tested for initial configurations of different lengths as defined in Table 4.1. The number of simultaneous active agents was varied by setting $\mu \in \{0.01, 0.05, 0.1, 0.2, 0.3\}$. A total of 100 trials were performed for each setting, with simulations stopping when 50 timesteps passed without the structure changing. Each trial was initialised from a different randomly-selected trial of the parallel cantilever construction algorithm with $\delta = 10$ timesteps and the appropriate link strengths when the structure reaches the specified length. Two trials with weak links, $L = 1.6$ m, $\mu = 0.3$ did not finish within a predefined time limit of 50 steps per agent in the initial bridge. The structures were reduced to 24 agents, but additional agents repeatedly released and replaced themselves in the same set of locations, resulting in the structure never settling for the required number of steps. They are excluded from the results below.

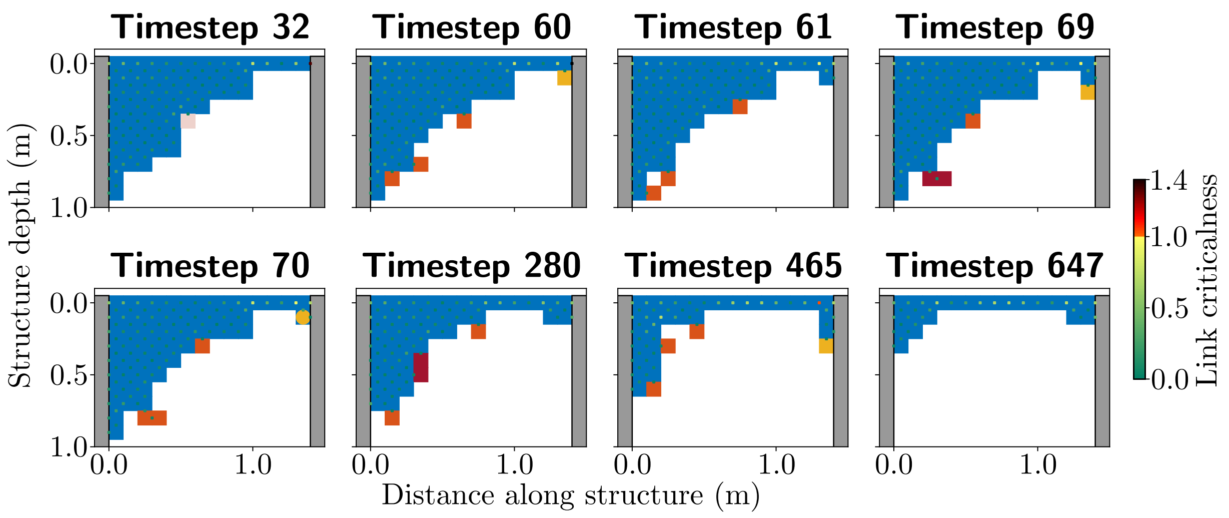
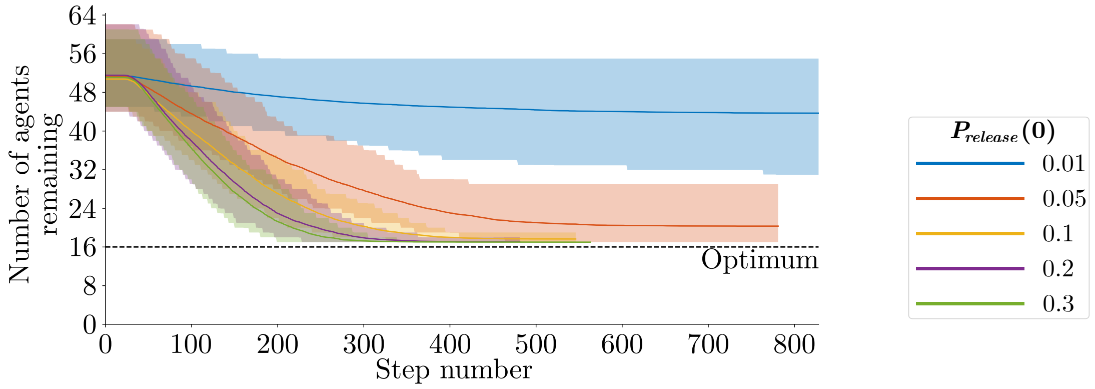


Figure 4.9. Frames from a bridge optimisation sequence for weak links, $P_{release}(0) = 0.05$. Placed agents are shown blue, and active agents are coloured by their mode: `releasing` (peach), `gathering` (orange), `placing` or `escaping` (yellow), and `swapping` (red). The fixed support is grey, and links are coloured by their criticalness.

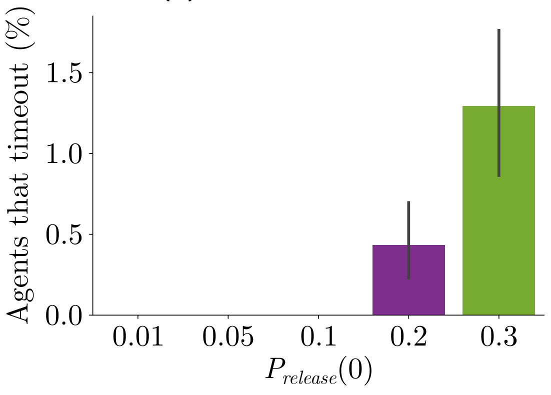
Figure 4.9 shows an example trial for weak links with $\mu = 0.05$. The trial begins with 62 placed agents, and it takes 32 timesteps for the first agent to probabilistically release. More agents release as this one gathers the necessary force information, so several swaps occur before the first agent reaches column L in timestep 60. The link attaching the structure to the right support is the only critical one measured, so this active agent calculates a high $p_{column}(L)$ and subsequently places in this column in timestep 61. This makes the structure stable, so the next agent to reach column L switches to the `escaping` mode and passes through the structure to the top, as can be seen in timesteps 69 and 70. The structure at timestep 280 shows agents in the `swapping` mode, and together with the snapshot at timestep 465 illustrates how agents are both added to and removed from the structure as the trial progresses: notice how the portion attached to the right support grows, shrinks, and changes shape as timesteps pass. The trial ends in timestep 647 when 22 agents remain, representing a 65 % reduction. For comparison, the optimal bridge of this length and link strength contains 18 agents, so the final structure contains only four more agents than is optimal.

The trial shown in Figure 4.9 is representative of the general trend observed across all trials. The link connecting the structure to the right support is often critical to begin with, but is quickly reinforced. Subsequent active agents leave the structure, occasionally forming queues, particularly around canyons. However, the algorithm is observed to avoid deadlocks in all trials. Over time, the number of simultaneous active agents decreases and the changes in the structure becomes more minor. The final configurations contain significantly fewer agents than the initial configurations.

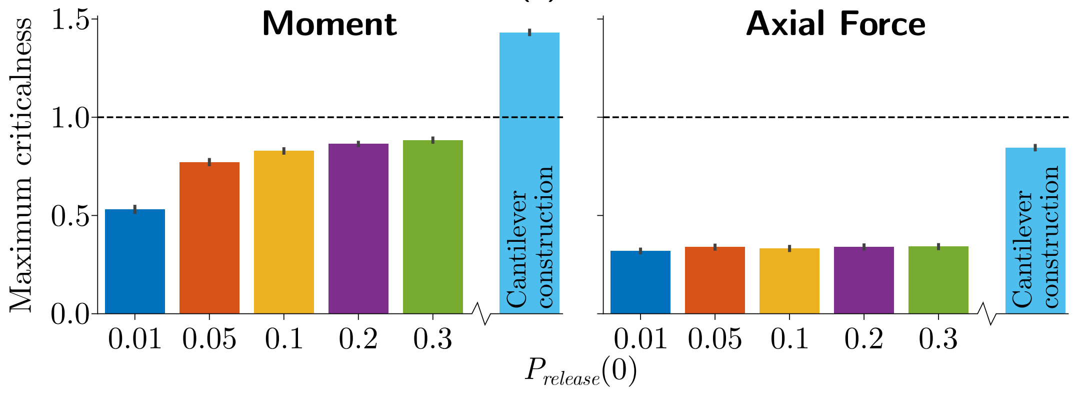
The average performance of the algorithm across the 100 trials for each situation is first compared across structures of the same length but different numbers of simultaneous active agents, controlled by the parameter μ. Figure 4.10 shows how varying this parameter affects the rate at which agents are removed. This figure is drawn for bridges of length 1.6 m with links of medium strength, but the behaviour is similar for all tested parameter



(a)



(b)



(c)

Figure 4.10. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.6 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included in cyan for comparison.

settings: see Appendix D for a complete set of graphs for all lengths and link strengths tested. Figure 4.10a shows that during each trial, the number of agents in the structure first decreases at a high rate, then slows down as the optimum number of agents is approached. When agents are more likely to release, the number of agents in the structure drops at a higher rate, but the effect decreases with larger μ. Another effect of this is that agents swap more often, increasing congestion around the structure as they interact. Agents therefore more frequently timeout before they can reach their intended placement location, whereupon they are either added to the structure or switch to the `escaping` mode depending on their location. However, Figure 4.10b shows that timeout is never a very common occurrence, especially compared to the rates observed during cantilever construction (Figure 3.17c).

The cost for the faster optimisation observed with larger μ is a higher maximum γ during construction, as shown in Figure 4.10c. This occurs as there are more agents moving around the structure without providing support to it at any time, thus the measured M and F values are typically higher than if there are fewer active agents moving at once. This means the force information the active agents receive does not accurately reflect the forces within the structure under only the weight of placed agents, and is also outdated by the time they reach a position to reinforce the structure. This figure also shows that the maximum γ during bridge optimisation is typically lower than during the initial cantilever construction. This means that, assuming the cantilever construction did not result in damage to agents within the structure, the bridge optimisation will probably not either. Finally, it can also be seen that γ^M is greater than γ^F during bridge optimisation, indicating the moment capacity within links guides the behaviour of the algorithm to a greater degree than the axial force capacity.

The average performance is now compared across structures of different lengths but with active agents initialising at the same rate. Figure 4.11 shows the results across different L for $\mu = 0.2$ as a representative example, while Appendix D contains full results for all values of μ tested. It can be seen in Figure 4.11a that the number of agents remaining after optimisation is significantly lower than the initial configuration for all parameter settings. The final number of agents is closer to the optimum for stronger and shorter structures. Figure 4.11b shows that the maximum γ during the optimisation is typically larger for structures that are weaker and longer. It also restates the trends seen in Figure 4.10c: the maximum γ is typically higher during cantilever construction than bridge optimisation, and γ^M is higher than γ^F.

4.5.1 Bridges in Use

The purpose of creating these self-assembled bridges is to allow agents to cross the gap to accomplish tasks. It would therefore be beneficial for the bridge optimisation procedure to be able to take place while other agents are crossing the bridge. This would allow more agents to reach the opposite side quicker, thus aiding in the completion of the tasks there.

The performance of the bridge optimisation algorithm while agents cross it was not tested systematically, but informal trials were performed. In these trials, active agents were added in position $(0, 0)$ at regular intervals of δ timesteps at the same time as the bridge optimisation algorithm operated. These additional agents crossed the bridge and exited the simulation at the other end.

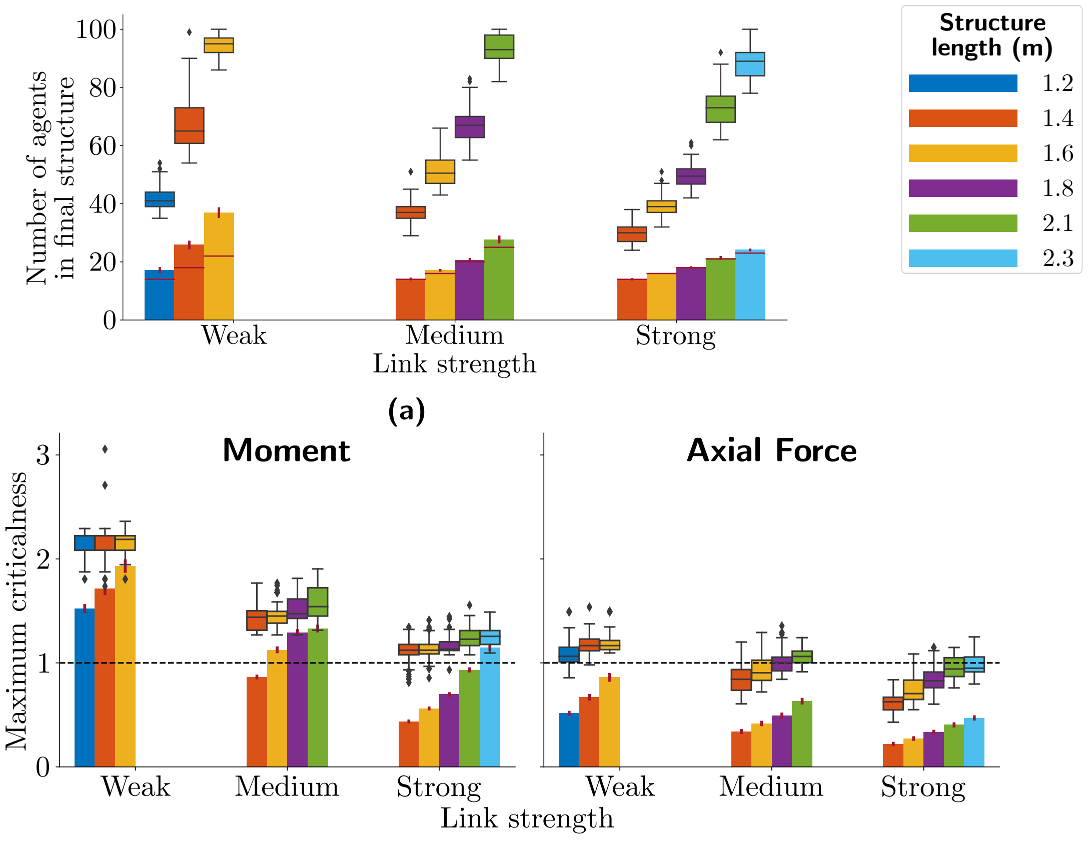


Figure 4.11. The effect of varying structure length on **(a)** the number of agents remaining in the bridge after optimisation, and **(b)** the maximum moment and axial force criticalness throughout the optimisation. Each bar shows the mean of 100 trials for $\mu = 0.2$, vertical red lines show the 95 % confidence intervals, and boxplots show the equivalent data during construction of the cantilevers used as the starting configurations. In **(a)**, the optimum numbers of agents for each bridge are shown by the horizontal red lines.

An example of a bridge of length $L = 1.4\,\text{m}$ with weak links being optimised using $\mu = 0.2$ while agents cross it $\delta = 4$ timesteps apart is given in Figure 4.12. It can be seen that the bridge optimisation algorithm takes 448 timesteps to reduce the number of agents in the bridge from 64 to 30, a reduction of 53 %. In this time, 108 other agents crossed the bridge and can begin work on the tasks on the opposite side.

The limited trials performed indicate that the algorithm is capable of functioning in this modified scenario, demonstrating its applicability in a range of situations. In this case, the total weight of active agents on top of the structure is at most WL, so they cannot exert loads on the bridge greater than the weight of the bridge itself. Further tests should be carried out to examine how the bridge optimisation algorithm performs if agents transport heavy objects across the bridge while the optimisation occurs.

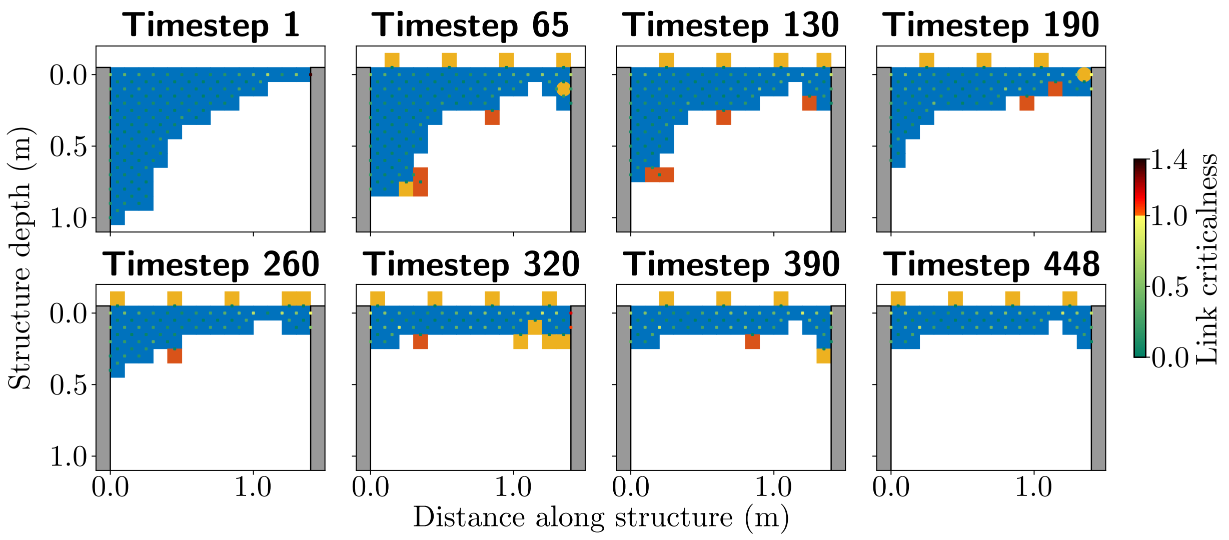


Figure 4.12. Frames from a bridge optimisation sequence for weak links, $P_{release}(0) = 0.2$, and active agents initialising in $(0, 0)$ to travel across the bridge every 4 timesteps. Placed agents are shown blue, and active agents are coloured by their mode: `releasing` (peach), `gathering` (orange), `placing` or `escaping` (yellow), and `swapping` (red). The fixed support is grey, and links are coloured by their criticalness.

4.6 Summary

This chapter has presented a distributed force-aware algorithm to optimise a self-assembled bridge across a void. The algorithm reduces the number of agents within the bridge, while taking into account local force information to ensure the structure will not collapse under its self-weight. This procedure releases more agents to accomplish different tasks on the other side of the void. Optimal bridges were also calculated to compare the performance of the algorithm against.

The algorithm was shown to remove the majority of the agents from initial configurations drawn from the cantilever construction algorithm of Chapter 3. The resulting bridges are near-optimal with respect to the number of agents required to safely span this gap. The maximum moment and axial force in links during this optimisation compares favourably with those that occurred during the construction of the initial bridge. Using fewer simultaneous active agents led to the algorithm creating bridges with fewer agents. This is because the active agents are able to make decisions about where to place based on more accurate information, and congestion is reduced so they can reach their intended placement locations. Longer structures and those with weaker links required more agents and experienced greater forces in their links throughout the operation of the algorithm. It was also demonstrated that the algorithm can allow for optimisation when agents are travelling across the bridge, but further trials are required to investigate the effect of greater loads crossing.

The next chapter considers the final stage in the lifespan of a bridge: how it should be deconstructed when it is no longer required. The bridge optimisation algorithm will be verified in real-life in Chapter 6, along with the other self-assembly algorithms developed in this thesis.

Chapter 5

Bridge Deconstruction

In the previous chapters, distributed force-aware self-assembly algorithms were presented that enable a group of robots to cross a gap by building a bridge across it from their bodies. These algorithms use local force measurements to inform the motion of robots and aim to make structures that will not collapse under gravity. The bridges are built by first extending a cantilever from one side of the void (Chapter 3), then optimised to reduce the number of agents in the structure (Chapter 4) to leave a bridge similar to that shown in Figure 5.1a.

This chapter considers how a self-assembled bridge can be safely dismantled when it is no longer required, referred to as *bridge deconstruction*. The agents must first convert the existing bridge to one that will be stable when supported on one side (Figure 5.1b), before the other side can be released and the structure dismantled (Figure 5.1c). The difficulty of the initial step is compounded as the guiding force measurements are from a structure supported on both sides, whereas the agents aim to construct a structure that will be stable when only supported on one side.

This chapter is organised as follows. A formal definition of the problem is given in Section 5.1, then the distributed bridge deconstruction algorithm is described in Section 5.2. The algorithm is verified in simulation in Section 5.3, and conclusions are drawn in Section 5.4.

The work in this chapter is based on sections of the author's published work [190], with certain aspects expanded upon. In particular, this chapter draws from Sections IV.B, IV.C, IV.D, IV.E, V.B, and V.D of this work.

5.1 Problem Formulation

The situation considered in this chapter is almost identical to that for bridge optimisation, described in Section 4.1. However, the bridge optimisation procedure always considered a structure supported at both ends, whereas during deconstruction one of the supports will eventually be released. After this, the restrictions placed on the structure are relaxed. The structure is no longer required to have a narrow section or an inflection point. In addition, the continuity condition is relaxed to require placed agents to be connected to just one of the fixed supports by a horizontal chain of placed agents, instead of both sides; they are still required to be connected to the placed agent at the top of their column by a vertical chain of placed agents, but the top of the column is not required to

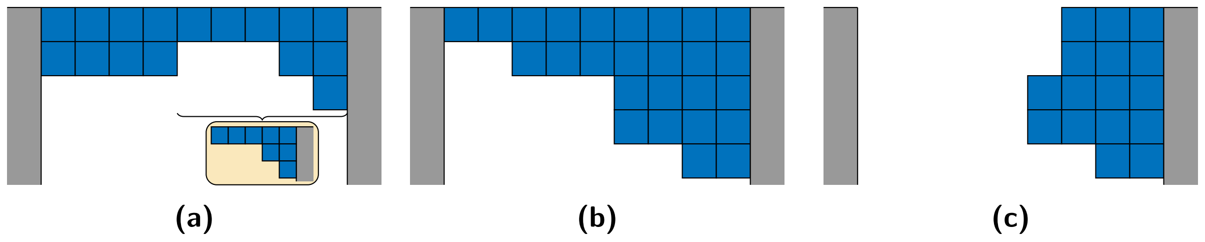


Figure 5.1. (a) The bridge deconstruction algorithm begins with a structure between two vertical fixed supports: the inset depicts the equivalent unsupported cantilever. **(b)** Agents are added to this structure so that it will be stable when one support is released, then **(c)** the agents are removed from the structure. Agents are shown in blue and the fixed supports in grey.

be in row 1. With these modifications, the structure in Figure 5.1c is continuous.

The bridge deconstruction algorithm dismantles an initial structure between the two vertical fixed supports and recovers agents on the opposite side of the void to where the initial cantilever extended from. In the case considered here, the cantilevers extended from the left, so the agents leave the structure above the right support. It is assumed that there is a continuous supply of agents above the latter support in position $(0, L+1)$ that have previously crossed the bridge and can add themselves to the structure. In addition, the agents already within the structure can reposition themselves as in bridge optimisation.

There is one more addition made to the scenario considered in this chapter: the active agents are able to measure the shear force S in their links, in addition to the moment M and axial force F. This is not used to calculate a criticalness or determine the stability of the structure. Rather, it is used to improve the estimate of the forces in the *equivalent unsupported cantilever*, as described in Section 5.2.2. This term refers to the narrow section and the columns to its right were the link on its left side to be released, leaving it only attached to the right support (Figure 5.1a).

The aim of the bridge deconstruction algorithm is twofold. Firstly, a structure that will not collapse when it is detached from the left support should be built. This is achieved by adding new agents to the structure and repositioning existing ones from left of the narrow section. In effect, the equivalent unsupported cantilever is extended towards the left support and reinforced until it is sufficiently strong. The second stage then begins: the structure is detached from the left support, whereupon agents are removed from the environment above the right support. The previous algorithms aimed to correct for unstable configurations as soon as they occur. Here, the agents are not concerned with the current state of the structure, but rather with building a structure that will be stable when it is released from the left support.

5.2 Algorithm Design

In this section, the bridge deconstruction algorithm is presented. The algorithm is first described in detail to demonstrate how the agents move and communicate with one another (Section 5.2.1). The method by which agents calculate what the forces would be in the equivalent unsupported cantilever is then explained (Section 5.2.2).

5.2.1 Algorithm Description

The bridge deconstruction algorithm is shown in Algorithm 5.1, highlighting the differences between it and the bridge optimisation algorithm (Algorithm 4.1) it is based on. Agents transition through broadly the same modes, but with the changes and additions described below. The deconstruction procedure consists of two phases:

1. **Reinforcement:** Agents are added to the structure from above the right support to produce a structure strong enough to not collapse when it disconnects from the left support (from Figure 5.1a to Figure 5.1b).

2. **Removal:** The structure disconnects from the left support and agents leave it above the right support (from Figure 5.1b to Figure 5.1c).

In addition to the `releasing`, `gathering`, `escaping`, `placing`, and `swapping` modes that exist for bridge optimisation, a further mode is introduced. This mode is called `force-releasing`, and describes placed agents that should release and become active regardless of γ in their links (Lines 2 – 6). Active agents may instruct adjacent placed agents to enter this mode to initiate the removal phase, extend the equivalent unsupported cantilever, or to deconstruct canyons as explained below and illustrated in Figure 5.2. When such an agent releases, they either enter the `gathering` or `escaping` mode, depending on the reason they were instructed to release: such agents are therefore described as `force-releasing` *to gather* or *to escape* respectively. If a `force-releasing` agent becoming active would violate the continuity condition, messages are instead passed down the column, taking one timestep per row, to switch the lower agents to `force-releasing` to gather (Line 6 and Figure 5.2a). Such columns thus deconstruct from the bottom.

Active agents can arise in two further ways. Firstly, placed agents on the lower perimeter and the left side of the inflection point with an empty cell on their right are release candidates as in the bridge optimisation algorithm: they release themselves with probability $P_{release}$ as before, with $\mu = 0.2$ (Lines 7 – 14). Agents on the right of the inflection point cannot do this, as it is assumed that they are already placed in a satisfactory position. Secondly, additional `gathering` active agents enter the simulation in position $(0, L)$ a fixed number of timesteps δ apart, assuming this location is unoccupied, until the first `escaping` agent occupies this cell. These agents then travel along the top of the structure, obtaining measurements of M and F in the links of the agents below them as they move (Line 16). They continue moving until they are above the leftmost column of the narrow section, which they are informed of by the agent at the top of this column. They pass through the structure to the lower perimeter at this point. Agents on the lower perimeter in the `gathering` mode immediately move in the direction of column L without first visiting column 0 (Line 17).

When a `gathering` agent on the lower perimeter reaches column L, it calculates a probability mass function $p_{column}(c)$ describing the probability of placing in each column (Line 18). This is done as described in Section 3.4.2.A, but instead of considering the current state of the structure, the $\gamma^{\alpha,\beta}$ arrays are modified to estimate what the forces would be within the equivalent unsupported cantilever. This is calculated by augmenting the recorded measurements as described in Section 5.2.2. The probability mass function is set to 0 left of the narrow section. Regardless of the force measurements received, the active agent switches to the `placing` mode (Line 23). The behaviour in this mode depends on what it believes the state of the equivalent unsupported cantilever would be:

Algorithm 5.1. The bridge deconstruction algorithm, emphasising differences with the bridge optimisation algorithm (Algorithm 4.1).

```
switch mode do
    case force-releasing do
        if Able to release then
            mode ← escaping;
        else if Continuity condition would be violated by release then
            Set agent below to force-releasing;
    case releasing do
        Record $M$ and $F$ from placed agent above into $\gamma_c^{\alpha,\beta}$;
        if Agent above has a critical link then
            Place back here (agent no longer active);
            return
        else
            mode ← gathering;
            Make step*;
    case gathering do
        Record $M$ and $F$ from placed agent above or below into $\gamma_c^{\alpha,\beta}$;
        if Agent in column $L$ then
            Calculate $p_{column}(c)$ from $\boldsymbol{\gamma}^{\alpha,\beta}$;
            if Equivalent unsupported cantilever would be stable then
                $c_{target}$ ← leftmost column in narrow section;
            else
                $c_{target}$ ← sample from $p_{column}(c)$ without replacement;
            mode ← placing;
        else
            Make step*;
    case escaping do
        if In position $(0, L)$ then
            Agent leaves simulation;
            return
        Set agent below to force-releasing to escape if required;
        Make step*;
    case placing do
        if In column $c_{target}$ then
            Attempt to place or set adjacent agent to force-releasing;
            if Placement succeeded then
                Agent no longer active;
                return
            else if Tip was previously released then
                mode ← escaping;
            else
                $c_{target}$ ← sample from $p_{column}(c)$ without replacement;
        Make step*;
    case swapping do
        mode ← previous mode;
```

Algorithm 5.1. [continued] The bridge deconstruction algorithm, emphasising differences with the bridge optimisation algorithm (Algorithm 4.1).

```
45 if In a canyon then
46     Set agent on left to force-releasing to gather;
47 if Agent above force-releasing to escape then
48     $c_{target} \leftarrow$ list increasing from column to the right;
49 if In position $(2, 1)$ and placed agent on right then
50     Set agent above to force-releasing to escape;
51 if In row 0 or agent on the right is force-releasing to escape then
52     mode $\leftarrow$ escaping;
53 if Agent stationary for $> \delta$ timesteps then
54     Attempt placement;
55     if Placement succeeded then
56         Agent no longer active;
* Steps made if the agent is not the sole support of another
```

- **Unstable:** The active agent draws c_{target} from $p_{column}(c)$ to provide reinforcement as before (Line 22). It then moves to this column and attempts to place here (Line 34).

- **Stable:** The active agent attempts to make the equivalent unsupported cantilever longer by first setting c_{target} to the leftmost column it believes to be in the narrow section (Line 20). When it reaches the top of this column it does not place here, but rather sets an adjacent agent to `force-releasing` (Line 34). If there is a placed agent to its left, it is switched to `force-releasing` to gather (Figure 5.2a). If instead $c_{target} = 1$ here, the active agent sets the agent above to `force-releasing` to escape, initiating the removal phase in the process (Figure 5.2b). In both cases, the active agent then draws another c_{target} from $p_{column}(c)$ without replacement to reinforce the structure (Line 41) and moves towards it.

As was the case for cantilever construction and bridge optimisation, agents draw a new value of c_{target} from $p_{column}(c)$ without replacement if it was not possible to place in the original c_{target} without violating the continuity condition (Line 41). If the agent reaches c_{target} and finds the tip was already released, it switches to the `escaping` mode and leaves the structure (Lines 38 – 39).

The `escaping` mode is modified to describe agents implementing the removal phase, where the structure is dismantled columnwise from the left. Before agents make their first step in this mode, they set the agent below them to `force-releasing` to escape (Line 30). They then travel up their column and right along the top of the structure to position $(0, L)$, where they exit the simulation. When the agent that was originally the lowest placed agent in column c reaches the top of column $c + 1$, it sets the agent at the top of this column to `force-releasing` to escape (Line 30 and Figure 5.2c).

As was the case during cantilever construction and bridge optimisation, agents timeout if they have been stationary for too many timesteps. Here the timeout period is set to

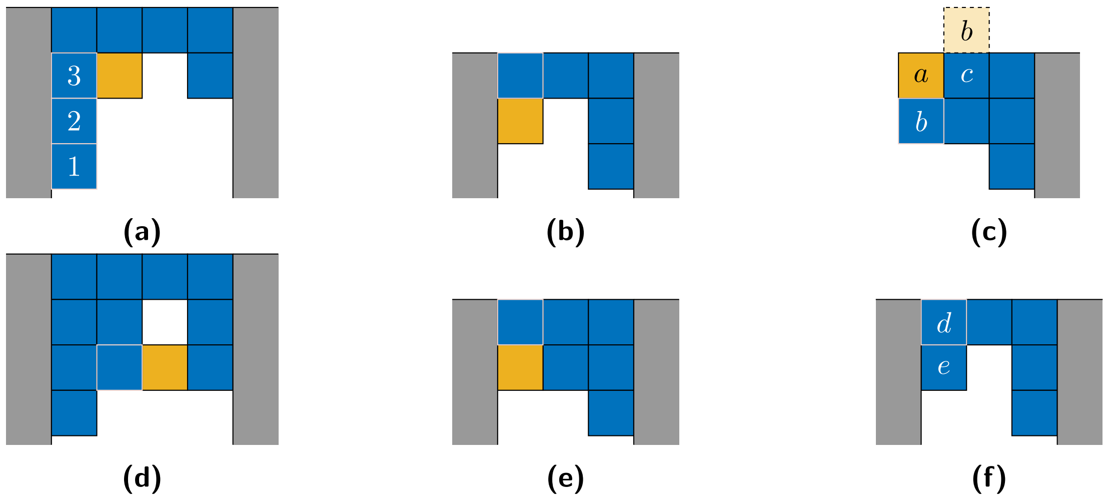


Figure 5.2. Situations in which agents enter the `force-releasing` mode. Placed agents are shown in blue, active agents in yellow, the fixed supports in grey, and `force-releasing` agents have a cream border. **(a)** The active agent believes the equivalent unsupported cantilever would be stable, so sets the agent left of the narrow section in row 2 to `force-releasing` to gather. However, this agent is blocked by others below them, so it sets these to `force-releasing` to gather as well: these agents will then release in the numbered order shown. **(b)** The active agent believes the equivalent unsupported cantilever would be stable, so sets the agent at the tip to `force-releasing` to escape, beginning the removal phase. **(c)** Agent a is `escaping`, so sets agent b to `force-releasing` to escape when it moves off; when agent b reaches the position shown paler, it will set agent c to `force-releasing` to escape. **(d)** The active agent is in a canyon, so sets the agent on its left to `force-releasing` to gather. **(e)** An active agent finishes advancing in position $(2, 1)$ with a placed agent to its right, so the agent at the tip is set to `force-releasing` to escape and the removal phase begins. **(f)** Agent d is `force-releasing` to escape, but is blocked by agent e below it: this agent is set to `force-releasing` to gather.

the number of timesteps between adding agents δ (Line 53). Before assessing whether an active agent should timeout, it checks the cells in its von Neumann neighbourhood to see if any additional behaviours should be triggered:

- If the active agent is inside a canyon, it will tell the agent on its left to enter the `force-releasing` mode to gather (Lines 45 – 46 and Figure 5.2d). This reduces congestion around canyons, and will also extend the equivalent unsupported cantilever if this occurs in row 2.

- If the active agent is below one that is `force-releasing` to escape, it switches to the `placing` mode and sequentially selects c_{target} to place in the closest column to its right that satisfies the continuity condition (Lines 47 – 48). The active agent therefore leaves the area below where the removal phase is occurring without obstructing other agents.

- If the active agent reaches position $(2, 1)$ and there is a placed agent to its right, it is assumed that the structure is unlikely to get any more stable. The active agent therefore initiates the removal phase (Lines 49 – 50 and Figure 5.2e).

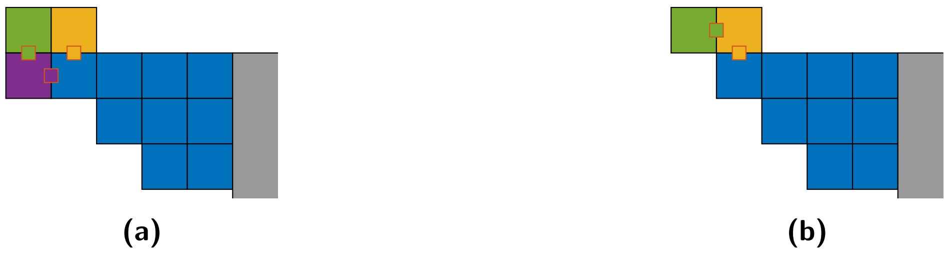


Figure 5.3. Setting active links when around the tip. Placed agents are shown in blue, fixed supports in grey, and active agents in other colours. The active link of each agent is shown by the small orange-bordered square of matching colour.

- If the active agent finishes advancing in row 1 or left of an agent that is `force-releasing` to escape, it swaps to the `escaping` mode (Lines 51 – 52). This condition is only met during the removal phase, so the active agent should leave the structure instead of continuing to reinforce it.

One final scenario considered is that of a placed agent that is not connected to the right side of the structure occurring below an agent that is *force-releasing* to escape. This can arise either due to timeout, or active agents stepping backwards into the left fixed support to avoid swapping as described in Section 5.2.1.B. In such situations, the placed agent will become `force-releasing` to gather first to maintain continuity (Figure 5.2f).

5.2.1.A Active Links

Active agents are again in control of a single active link, set in the same manner as during bridge optimisation (Section 4.4.2.A). However, during deconstruction the choice of active link when agents are around the unsupported tip also needs to be explicitly set. The active link of the active agent at the unsupported tip is set to the right face (Figure 5.3a, purple). The active link of agents in row 0 is usually the bottom face, but will be the right face if no agent is below it (Figure 5.3, green). Since the existence of the unsupported tip means that the removal phase has begun, these agents should all switch to the `escaping` mode. Choosing active links in this way facilitates agents in row 0 to move right if possible, thus removing them from the structure.

5.2.1.B Swapping

Active agents travelling in opposite directions that encounter each other will usually exchange information and ‘become’ one another. During bridge optimisation, there are certain scenarios during which they remain stationary or temporarily change direction to make space for other agents to move into, as described in Section 4.4.2.B. These behaviours also apply during bridge deconstruction, with the following additions:

- **Escaping:** If an agent attempts to swap with an `escaping` agent, it will also switch to the `escaping` mode. As explained above, any agents at the unsupported tip will be in the `escaping` mode. Therefore, if an agent is attempting to travel down from row 0 and attempts to swap, it will instead enter the `escaping` mode and thus change direction (Figure 5.4a).

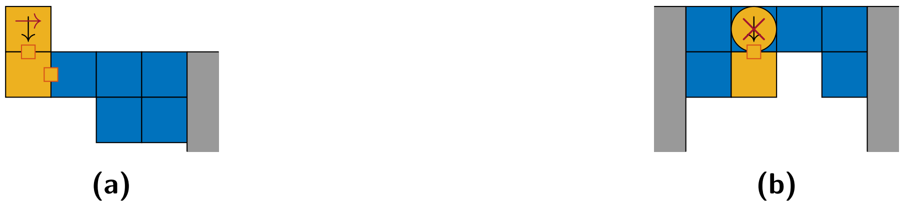


Figure 5.4. Special behaviours when agents should not swap information with each other. Active agents are shown in yellow (as a circle if they are passing through the structure), placed agents in blue, and the fixed supports in grey. Active links are shown as small yellow squares with orange borders. Salient directions of travel are shown by black arrows, and overridden directions are shown in red.

- **Passing through:** As in bridge optimisation, active agents passing through the structure that are blocked by another agent wait for this space to clear instead of swapping. During bridge deconstruction, agents pass through the structure from top to bottom instead of bottom to top, so the situation is as shown in Figure 5.4b.
- **Within canyons:** During bridge optimisation, if two agents are attempting to swap in a canyon (Figures 4.8d – 4.8f) and the lower one is `placing` with c_{target} set to this column, this agent will draw another c_{target} from $p_{column}(c)$ as placing at the top of a canyon would violate the requirement for a narrow section. During deconstruction, this swap is allowed to continue if the lower agent believes the structure is stable, as it will not place itself at the top of the canyon, thus the narrow section will be maintained. Otherwise, $p_{column}(c)$ is scaled to be 0 in and left of the canyon as placement should reinforce the equivalent unsupported cantilever. Another c_{target} is then drawn.

5.2.2 Forces in the Equivalent Unsupported Cantilever

During deconstruction, active agents are informed about the force distribution in the structure when it is supported at both ends. However, these agents are ultimately trying to build a cantilever that will only be supported from the right side. They should therefore place themselves so as to reduce what the forces would be in the equivalent unsupported cantilever. This requires each agent to convert the forces measured by placed agents in the bridge into what they would be in this cantilever. It would be possible to calculate this by having each agent model the structure as a truss as in the simulator, but this typically requires eigenvalue calculation, thus is algorithmically complex and intractable for the low-powered processors commonly found on modular robotic hardware, especially for large structures.

A faster calculation can be made by each agent using the principle of superposition [188]. The bridge in Figure 5.5a can be split on the left face of column 2 to give only the agents in the equivalent unsupported cantilever (shown darker), which are loaded as shown in case ① of Figure 5.5b. Agents measure M and F within links of these agents under this loading configuration, but would like to obtain M and F under gravity (loading case ②). This can be approximated by subtracting the force distributions under loading cases ③ – ⑤ from case ①.

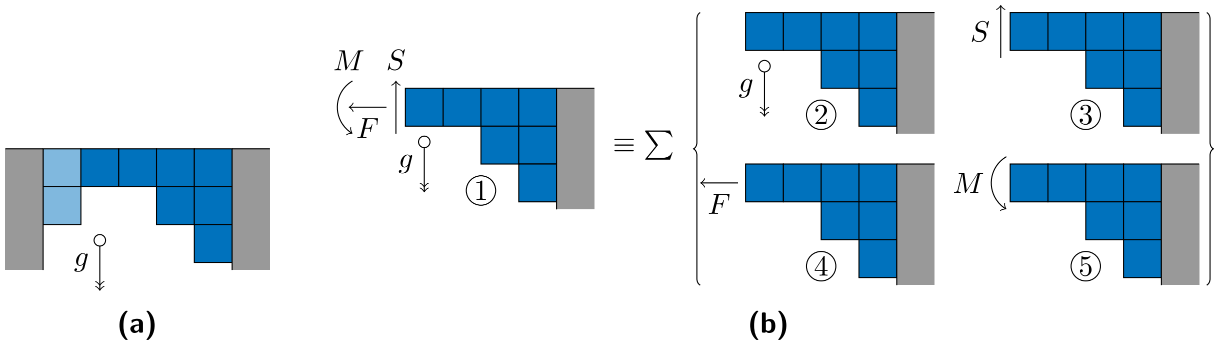


Figure 5.5. **(a)** Cutting off the columns to the left of the narrow section (shown paler) reveals the internal shear force S, axial force F, and moment M in the equivalent unsupported cantilever. This represents case ① of **(b)**, which can be decomposed into the sum of the separate loading cases ② – ⑤. The double-headed arrow g denotes the acceleration due to gravity, and the fixed supports are shown in grey.

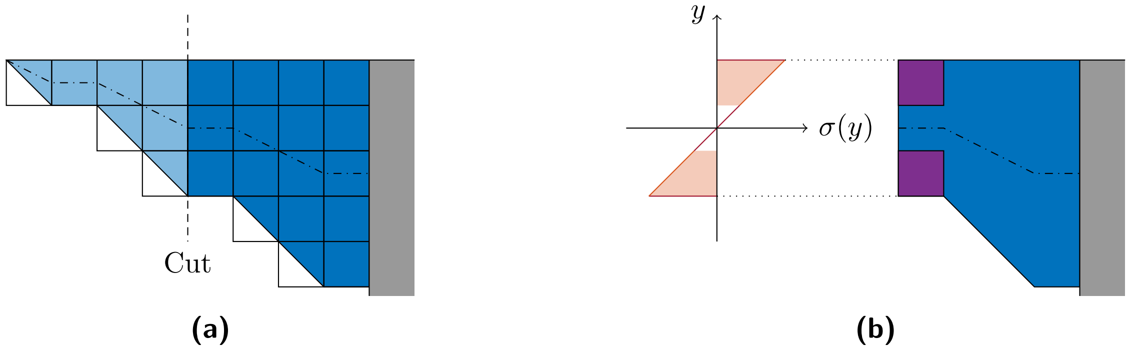


Figure 5.6. Approximating the force distribution due to an arbitrary known loading (not shown), with the fixed support shown in grey. **(a)** The outlines of a collection of agents arranged as a cantilever, where the filled region denotes the cantilever profile used to calculate the approximate force distribution. **(b)** A cut through this cantilever to remove the paler portion reveals the internal longitudinal stress distribution σ a distance y from the neutral axis (the dash-dotted line, assumed to be at the centroid of the cross-section). The highlighted regions of this distribution are used to calculate the equivalent M and F on the left faces of the agents shown in purple.

The active agent is informed of measurements of S, F, and M made by the placed agent at the left of the narrow section. Therefore the stress distributions under these additional loading configurations can be approximated by elastic beam theory [188] once the active agent reaches column L, having obtained all the available force information and measured the heights of each column. The structure is modelled as a cantilever whose height varies continually between these known heights (Figure 5.6a), and is analysed using elastic beam theory to calculate the distributions of the longitudinal stress σ across the faces of each column of agents under loading cases ③, ④, and ⑤. Equivalent M and F in the links of the top and bottom agents in each face are calculated from these distributions (Figure 5.6b).

The detailed mathematical analysis of these cases is given in Appendix C. It is shown that, for these loads, σ is a polynomial function of the geometric parameters of the structure and the known loads S, F, and M. The calculation of approximated forces in the relevant loading case ② is therefore quick to make. Note that forces can only be superimposed like this for linear elastic materials deflecting a small amount [188], and that these calculations are based on a simplified case of the stress distribution for a structure that is similar but not identical to the actual equivalent unsupported cantilever. The values will therefore only be approximate, but are still useful in improving $p_{column}(c)$ so that agents place in columns more suitable to reinforce the equivalent unsupported cantilever.

5.3 Simulation Results

The bridge deconstruction algorithm was tested for the same link strengths as the cantilever construction and bridge optimisation algorithms, termed weak, medium, and strong (Section 3.3.2). Trials began with a bridge chosen at random from the final structures of the bridge optimisation algorithm with $\mu = 0.2$. For each link strength, trials were performed for bridges of the same lengths that the bridge optimisation algorithm was tested for (Table 4.1), and the number of active agents in the simulation at a time was varied by setting $\delta \in \{6, 8, 10, 12\}$ timesteps. A total of 100 trials were performed for each combination of link strength, length, and δ. Each trial began with the resulting structure from a different random trial of the bridge optimisation algorithm for $\mu = 0.2$. Simulations were halted when either no agents remained, 50 timesteps had passed without the structure changing, or 200 agents had been added to the structure.

This section begins by presenting the results of the trials described above (Section 5.3.1). After this, Section 5.3.2 evaluates the effectiveness of the elastic beam theory calculations at approximating the forces in the equivalent unsupported cantilever as described in Section 5.2.2; the benefits this approximation brings to the deconstruction algorithm are also presented. Finally, Section 5.3.3 discusses a possible extension to this algorithm to enable deconstruction of non-optimised bridges without adding any additional agents.

5.3.1 Bridge Deconstruction Algorithm Performance

An example construction sequence is given in Figure 5.7 for links of medium strength. The initial structure is 2.1 m long, and has small buttresses at both ends. The first active agent initialises in timestep 1, and passes through to the lower perimeter of the structure in timestep 19, doing so in column 4 as this is at the left end of the narrow section. When it switches to the `placing` mode, it believes that the equivalent unsupported cantilever would be unstable, so places in position $(2, 19)$ in timestep 41 to provide support. The placed agent in position $(2, 3)$ also attempts to release itself in this timestep, but this causes the agent above to have a critical link, so it places back where it was in the next timestep. This agent tries again in timestep 151, and this time the additional support provided by the placed agents on the right of the structure means that the link of the agent above is no longer critical, so this agent becomes active in the `gathering` mode in timestep 152. The reinforcement phase continues, with more agents placing on the right side of the inflection point. In timestep 515, the active agent in position $(2, 1)$ has decided that the equivalent unsupported cantilever would be stable, so tells the agent at the tip to release in the subsequent timestep. This agent becomes active in the next

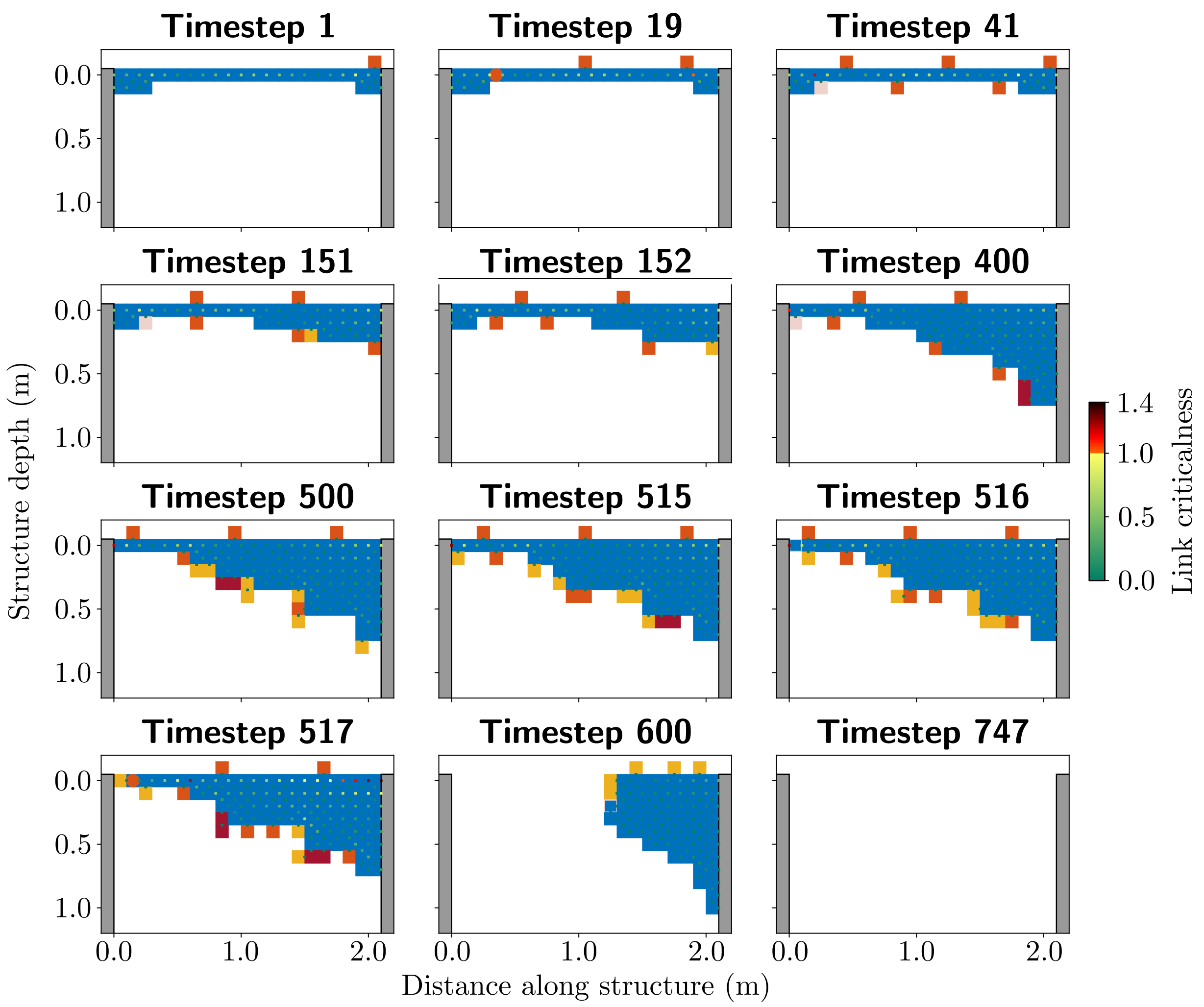


Figure 5.7. Frames from bridge deconstruction sequence with links of medium strength, $\delta = 8$ timesteps. Placed agents are shown blue, and active agents are coloured by their mode: `releasing` (peach), `gathering` (orange), `placing` or `escaping` (yellow), and `swapping` (red). Agents that are `force-releasing` are shown blue with a peach border. The fixed support is grey, and links are coloured by their criticalness.

timestep, and therefore begins the removal phase. The release causes more links near the right support to become critical, but as the removal phase progresses these links quickly cease to be critical. Note also how in timestep 517 an active agent passes through the structure in column 2 as it erroneously believes this is the left end of the narrow section due the randomised order in which agents move in each timestep. During the removal phase, agents leaving the structure are either adjacent or separated by up to two spaces, depending on the order in which they update each timestep. The final agent leaves the simulation in timestep 747, therefore the trial terminates at this point.

Figure 5.7 exemplifies the trends observed in all trials. Active agents place themselves in locations that reduce γ in the equivalent unsupported cantilever, although these locations may not be the same ones that reduce γ in the current bridge structure. The highest γ is observed immediately before and after the structure is released from the left support, but it is quickly reduced as agents leave the structure in the removal phase. During this phase, there is a constant flow of agents exiting the simulation at a high rate.

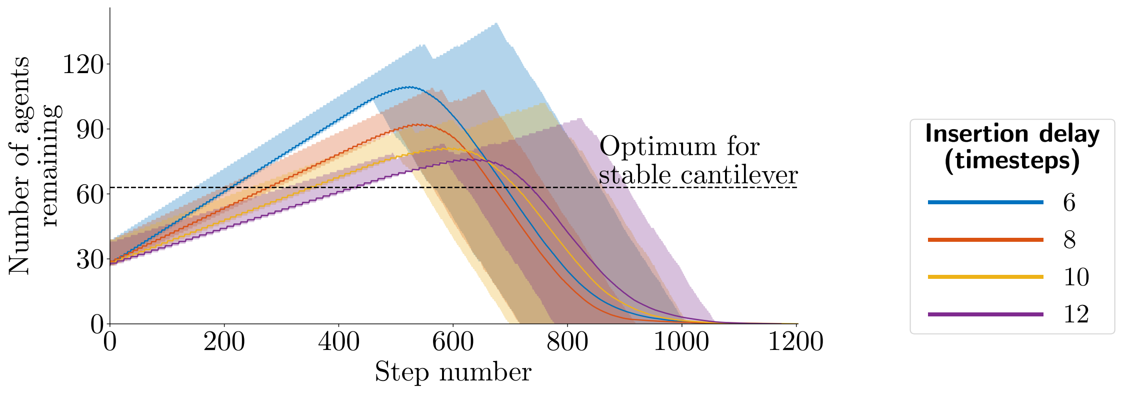
As for the bridge optimisation algorithm, the average performance of the bridge deconstruction algorithm across the 100 trials for each combination of link strength, length, and δ is first compared across structures of the same length but different numbers of simultaneous active agents, in this case controlled by adjusting δ. The effect of varying this parameter for links of medium strength with $L = 2.1\,\mathrm{m}$ is shown in Figure 5.8, while the results for different link strengths and L are given in Appendix E. Figure 5.8a clearly shows the reinforcement and removal phases as the regions where the number of agents in the simulation increases and decreases respectively. Lower δ results in more agents being added to the simulation at a faster rate, so they are unable to place in good locations due to congestion, meaning more agents are required before the removal phase begins. However, the switch to the removal phase occurs earlier in the simulation as the sheer number of agents means a stable structure is built faster. Since agents leave the structure at the same rate regardless of δ, the overall number of timesteps taken for the deconstruction algorithm to terminate is similar for all δ tested. More agents were required during the reinforcement phase than in the optimal cantilever of this length, but this discrepancy decreases with higher δ.

Figure 5.8b shows the percentage of agents in the structure when the removal phase begins that placed due to timeout: similar to the cantilever construction and bridge optimisation algorithms, the congestion that a greater number of simultaneous active agents incurs causes a higher proportion of agents to place due to timeout. The percentage of agents that timeout is similar to during cantilever construction, and higher than in bridge optimisation.

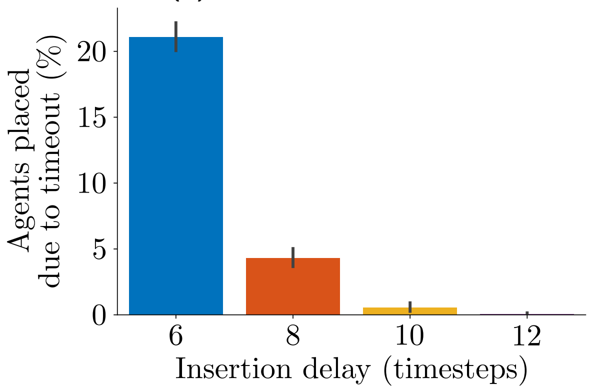
It is also interesting to compare the maximum γ throughout the operation of the algorithm. Figure 5.8c shows this maximum γ is not greatly affected by the choice of δ, and is usually comparable to the maximum that occurred during prior cantilever construction and bridge optimisation stages. Both γ^M and γ^F are of similar magnitude, indicating they are both important during deconstruction.

Comparisons are also made across different L and link strengths, shown in Figure 5.9 for $\delta = 10$ timesteps and for other δ in Appendix E. The number of agents in the structure when the removal phase begins is plotted in Figure 5.9a, illustrating how structures that are longer or have weaker links require more agents to deconstruct. The number of agents required is greater than in both the optimal cantilever and the original bridge, but similar to that in the original cantilever.

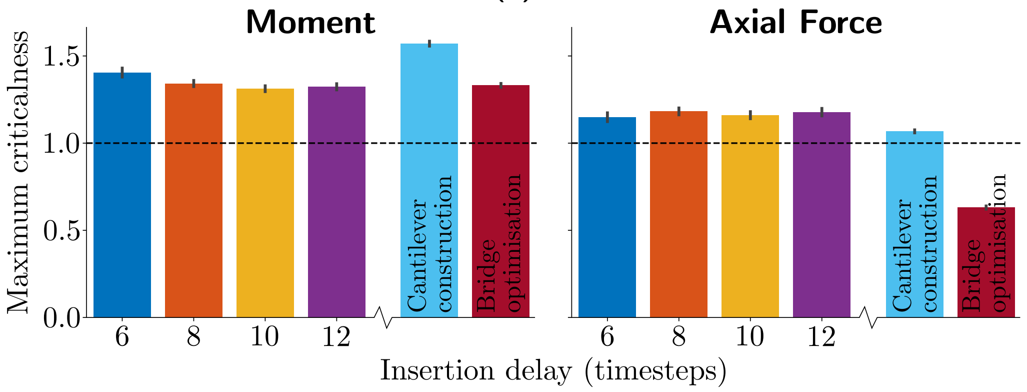
Figure 5.9b shows the maximum γ during deconstruction with $\delta = 10$ timesteps, which allows for comparison between the two phases and against the prior cantilever construction and bridge optimisation stages. The maximum γ during deconstruction is higher for longer structures and those with weaker links, and is usually similar or less than that incurred during prior construction stages. The largest γ^M typically occurs during the reinforcement phase, indicating that the agents successfully place themselves in locations that will reduce γ^M in the equivalent unsupported cantilever, but that these locations are not necessarily good for reducing γ^M at that instant. In contrast, the maximum γ^F always occurs during the removal phase, indicating that agents do not place themselves to efficiently reduce γ^F in the equivalent unsupported cantilever. This high γ^F normally occurs in the top right corner of the structure, as it does in timestep 517 of Figure 5.7. These behaviours can be explained as the largest errors in the approximated M and F in



(a)

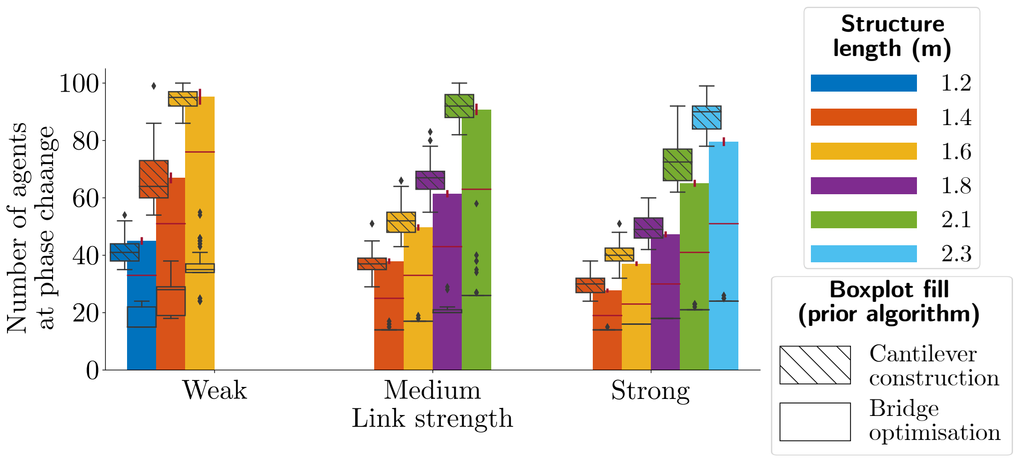


(b)

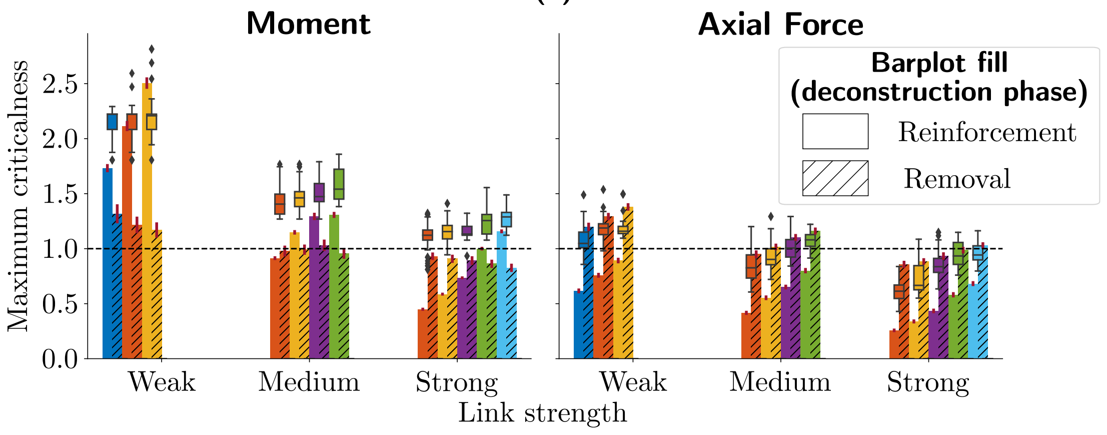


(c)

Figure 5.8. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 2.1 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison in cyan and red respectively.



(a)



(b)

Figure 5.9. The effect of varying structure length on **(a)** the number of agents in the structure when the removal phase begins, and **(b)** the maximum moment and axial force criticalness throughout the deconstruction, showing the differences in the reinforcement and removal phases. Each bar shows the mean of 100 trials for $\delta = 10$ timesteps, and vertical red lines show the 95 % confidence intervals. In **(a)** the optimum number of agents in a stable cantilever of this length is shown by the horizontal red line, and the boxplots show the number of agents in the prior cantilever construction and bridge optimisation stages separately. In **(b)**, the boxplots show the maximum criticalnesses during the prior cantilever construction and bridge optimisation stages combined.

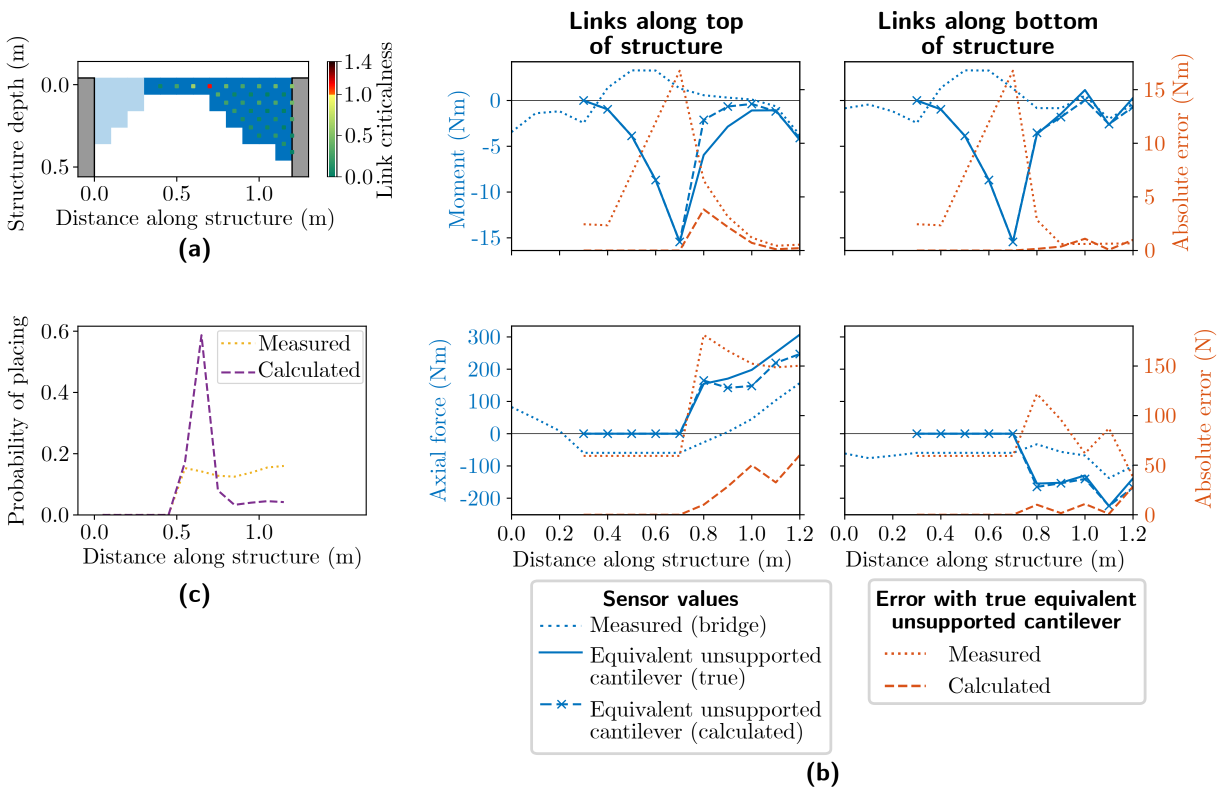


Figure 5.10. **(a)** An example bridge highlighting the region corresponding to its equivalent unsupported cantilever (darker): links are weak and coloured by criticalness in the equivalent unsupported cantilever. **(b)** The measured values of M and F in row links along the top and bottom of this structure are compared to the true values that would be measured in the equivalent unsupported cantilever, and to the values calculated by superposition. **(c)** The probability of placing in each column as calculated from the measured values and from the values calculated by superposition.

the equivalent unsupported cantilever are seen for F in links along the top of the structure close to the right support (Figure 5.10). These errors mean that agents do not place in locations that effectively reduce γ^F in this region.

5.3.2 Forces in the Equivalent Unsupported Cantilever

An example showing how the forces in the equivalent unsupported cantilever are approximated using superposition is given in Figure 5.10. The structure is shown in Figure 5.10a, and Figure 5.10b shows the values of M and F in row links along the top and bottom of the structure. The raw values measured by the agents (dotted lines) should be modified to obtain what they would be in the equivalent unsupported cantilever (solid lines). The values calculated using the superposition approximation (dashed lines with crosses) are seen to be close to the true values in the equivalent unsupported cantilever. The correction is not perfect, but the difference (shown in red) with the values in the equivalent unsupported cantilever is significantly reduced in almost all links compared to using the raw measurements from the bridge.

The probability an active agent would calculate of placing in each column is also plotted for weak links in Figure 5.10c. The link on the right of column 7 is not critical in the bridge configuration, but would be critical in the equivalent unsupported cantilever. When the correction is made, there is a high probability of placing here, which would reduce γ in this link in the equivalent unsupported cantilever. Without the correction, the probability distribution is much flatter, meaning it is much harder for agents to differentiate between columns that it would be effective to place in or not.

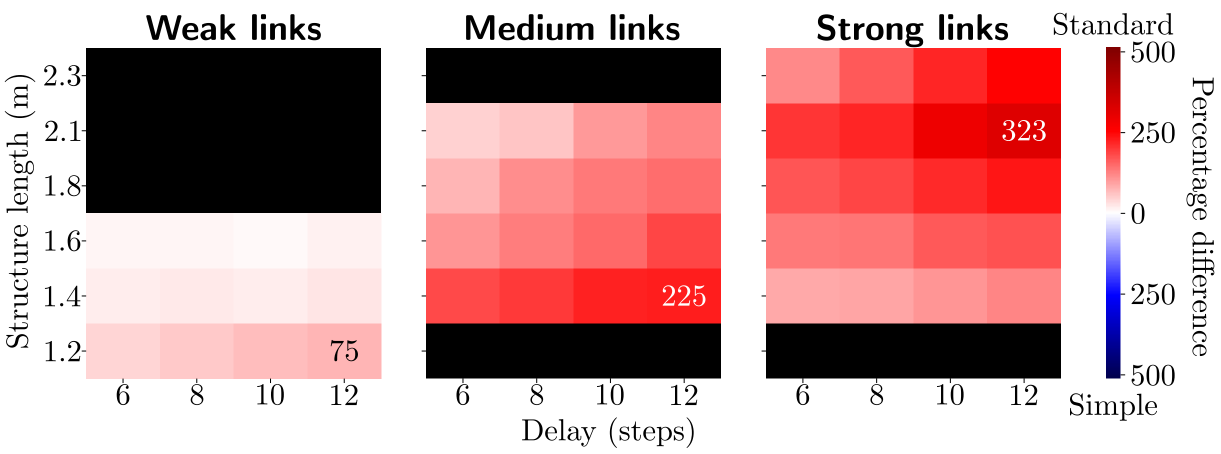
To further demonstrate the benefits of this calculation, the deconstruction algorithm was implemented using just the raw sensor readings, without the superposition correction. This is called the *simple variant* to differentiate it from the standard deconstruction algorithm. As for the standard algorithm, 100 trials of the same link strengths, structure lengths, and δ were performed. The results are summarised in Figure 5.11, which shows that the simple variant is able to deconstruct structures up to 323 % faster than the standard deconstruction algorithm, but at a cost of a significantly increased maximum γ in the structure. The maximum γ^M was up to 512 % greater for the simple variant, so structures are much more likely to break.

The reason for the simple variant giving faster deconstruction but higher γ is illustrated by the example in Figure 5.12a. Strong links are used, which are able to sustain an optimised bridge of length 1.8 m a single agent thick along its length. The first active agent switches to the `placing` mode in timestep 38. Since none of the links it measured were critical, it selects to release the left support. This is done in timestep 62, and immediately M in the link at the right support jumps to 5049 N m, corresponding to $\gamma^M = 5.8$. This is significantly higher than occurred during the initial construction, and is likely to cause this link to break. It is only in timestep 87 that this link ceases to be critical, so there is a long period of time in which this unstable structure must be sustained, increasing the likelihood of failure. In the standard deconstruction algorithm this would not happen: the agent calculates that the equivalent unsupported cantilever would be unstable, and so begins to reinforce the structure at the right support.

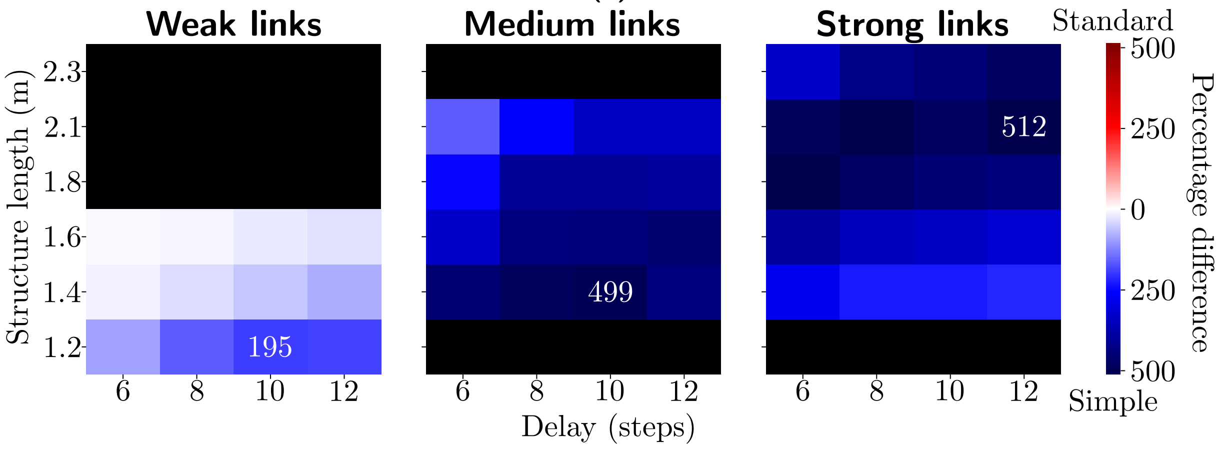
Another point to note is that the simple variant failed to finish on one occasion: it was deemed that construction had stalled as 200 agents were added to the simulation. Stages of this construction are shown in Figure 5.12b. Construction progresses reasonably well to begin with, as agents construct a buttress against the right support. However, at timestep 620 it can be seen that the structure has become inefficient, with a large square region beginning in column 6. This does not provide a large degree of support, but does increase the weight of the structure. This arises as agents are not able to accurately determine how best to place in the structure. The simulation terminates in timestep 931 as the 200th agent is added. The resulting structure is very tall, but still has a critical link right of column 3 that prevents the removal phase from initiating. This trial was repeated so 100 successful trials of the simple variant could be included to compare to the standard algorithm.

5.3.3 Deconstruction of Non-Optimised Bridges

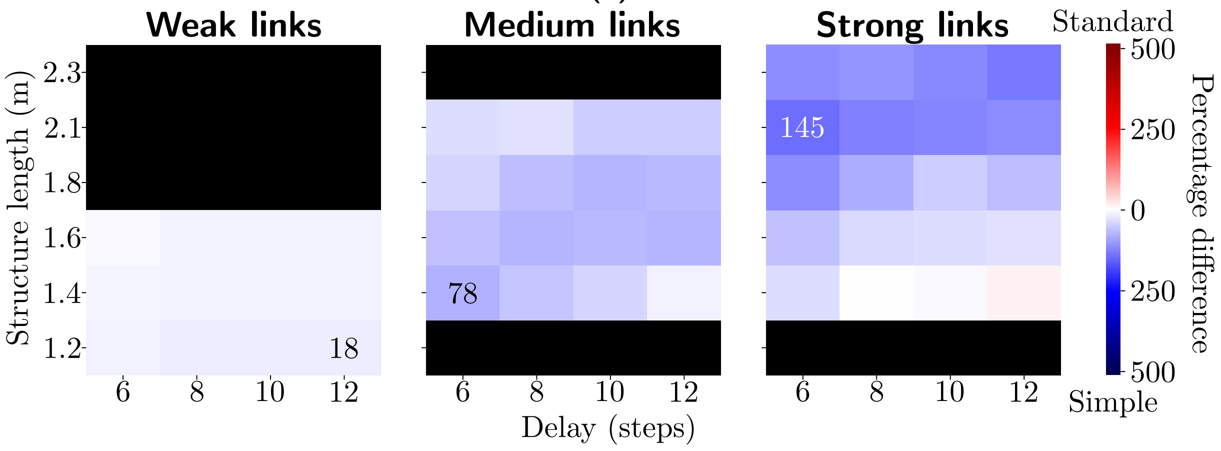
In the results presented above, the initial bridges were taken from trials of the bridge optimisation algorithm, and additional agents were added to the structure to make it stable before the removal phase. Another approach was also informally tested, where the initial bridges were taken from trials of the cantilever construction algorithm when they



(a)

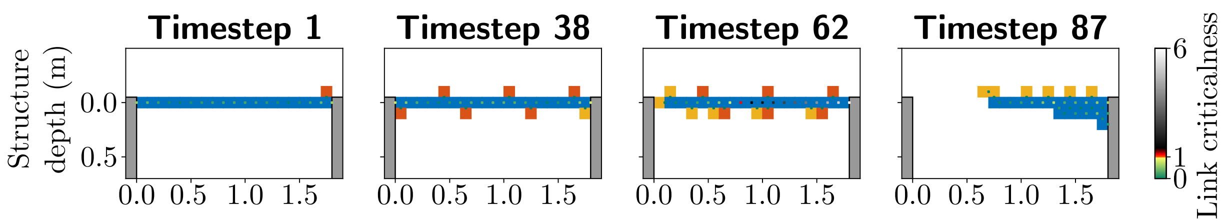


(b)

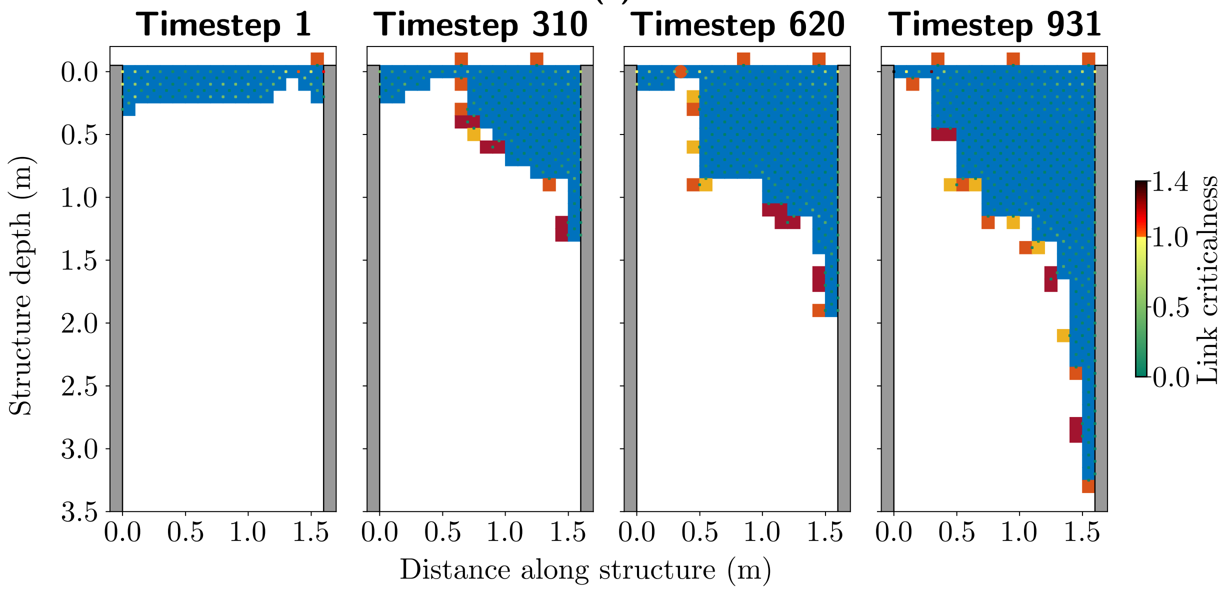


(c)

Figure 5.11. A comparison of the simple variant of the deconstruction algorithm to the standard version. Heatmaps show the percentage difference between the average values over 100 trials recorded by the deconstruction algorithms relative to the algorithm with the lower value. Black squares indicate trials that were not carried out. **(a)** The number of timesteps taken for deconstruction to finish. **(b & c)** The maximum γ^M and γ^F respectively throughout the whole structure during deconstruction. The largest value in each heatmap is explicitly stated.



(a)



(b)

Figure 5.12. Frames from bridge deconstruction sequences produced by the simple variant. **(a)** A trial with strong links, $\delta = 6$ timesteps, and **(b)** a trial with weak links, $\delta = 6$ timesteps. Placed agents are shown blue, and active agents are coloured by their mode: `gathering` (orange), `placing` or `escaping` (yellow), and `swapping` (red). The fixed support is grey, and links are coloured by their criticalness: the same colour scale is used as in previous figures, but it is extended in **(a)** to fade to white above a criticalness of 1.4 to highlight the extreme criticalnesses that occur when using the simple variant.

reached specified lengths. In these trials, it was assumed that all the available agents were used to construct this initial bridge, so the deconstruction algorithm was not able to add any additional agents to the structure. Instead, only the existing agents reconfigured themselves following the deconstruction algorithm.

An example trial demonstrating this situation is shown in Figure 5.13. Links are weak in this trial, and $\mu = 0.3$ so placed agents are more likely to become active than in the systematic trials above. Active agents are able to successfully move from the left to the right of the narrow section in preparation for the release of the left support. However, it can be seen in timestep 388 that releasing the agent in position $(2, 1)$ causes the remaining link attached to the left support to become critical, thus this agent will replace itself and not be released (Lines 9 – 11 of Algorithm 5.1). This means that the algorithm will never enter the removal phase: even if there are enough agents to build a stable structure, the placed agents on the right support are not permitted to reconfigure themselves repeatedly

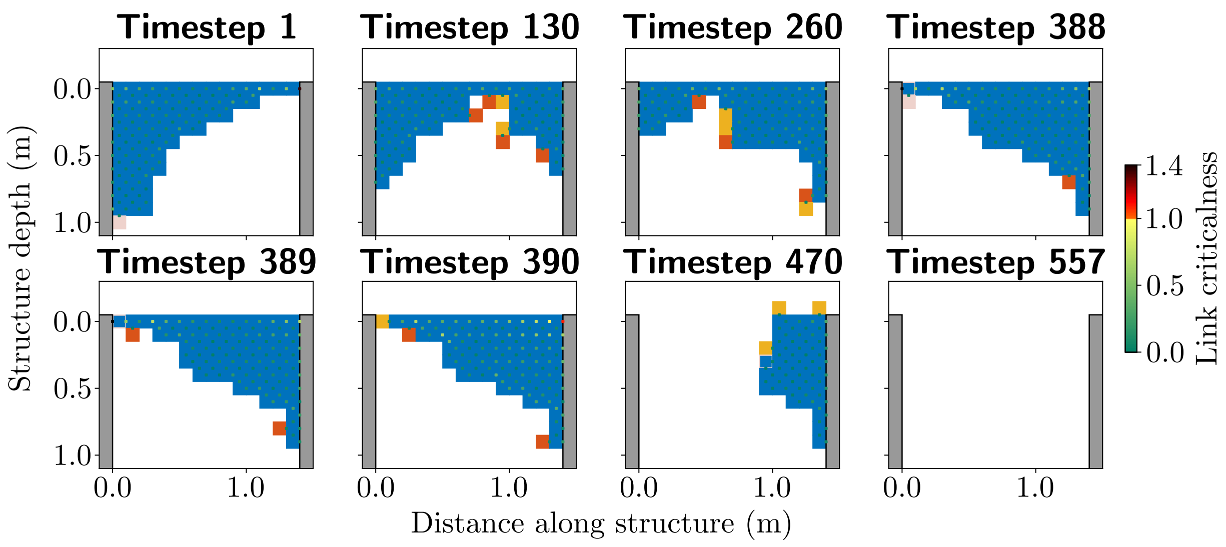


Figure 5.13. Frames from bridge deconstruction sequence with weak links from a non-optimised bridge with $P_{release}(0) = 0.3$. Placed agents are shown blue, and active agents are coloured by their mode: `releasing` (peach), `gathering` (orange), `placing` or `escaping` (yellow), and `swapping` (red). Agents that are `force-releasing` are shown blue with a peach border. The fixed support is grey, and links are coloured by their criticalness.

until this structure is built.

To escape this deadlock, the algorithm is modified in two ways. Firstly, the agent in position $(2, 1)$ is allowed to release itself even though the agent above has critical links, which is usually prohibited. This is particularly important as the link connecting the structure to the left support will often be critical due to the deflection of the structure, as explored in Section 4.2. The behaviour of the agent in position $(1, 1)$ is also modified to release itself and thus begin the removal phase when the agent below it moves. This is necessary as it is more likely that the equivalent unsupported cantilever will never be stable in this scenario compared to the case when agents could be added to the deconstructing bridge. It was observed that the deconstruction could usually proceed to this point without modifying the algorithm, but this final agent often measured the equivalent unsupported cantilever to be unstable, therefore it placed itself and the algorithm terminated without initiation of the removal phase. This heuristic modification allowed successful deconstruction to occur in all scenarios tested.

These changes can be seen in timesteps 388 – 390 of Figure 5.13. They enable the removal phase to begin, so the deconstruction can finish in timestep 557. This demonstrates the flexibility of the algorithm, but further trials are required to quantify the reliability and effect of the modifications described above. In particular, the rule to allow the agent in position $(2, 1)$ to release itself when the agent above has critical links may need to be extended to other agents to avoid deadlocks in similar scenarios.

5.4 Summary

In this chapter, the deconstruction of a bridge of self-assembled robots between two vertical fixed supports was considered. The bridge deconstruction algorithm achieved this in a distributed manner, leaving all agents on the opposite side of the void to where they began. As with the cantilever construction and bridge optimisation algorithms, this algorithm incorporates local force information to guide the construction, but these forces are augmented with elastic beam theory to build a structure that will be stable when one support is released.

The algorithm is capable of successfully deconstructing all the structures tested, usually requiring a similar number of agents to the initial bridge between the supports before optimisation. The maximum moment and axial force experienced during bridge deconstruction was typically greater than that during the prior bridge optimisation stage, but comparable to that recorded in the initial cantilever construction. Increasing δ leads to the algorithm operating with fewer simultaneous active agents, which means fewer agents are required to build a structure that can safely detach from the left support. However, the total number of timesteps required for the deconstruction to complete, and the maximum γ during this process, are largely unaffected by choice of δ. The algorithm was also shown to be able to deconstruct bridges without adding any additional agents, but further testing is required to investigate this ability in detail.

Algorithms have now been presented and tested in simulation to achieve the construction, optimisation, and deconstruction of bridges by self-assembling robots. In the following chapter, a hardware platform is developed and used to demonstrate these algorithms in real-life.

Chapter 6

Real-World Implementation

The algorithms for cantilever construction, bridge optimisation, and bridge deconstruction developed in the previous chapters were developed and verified using a custom Python simulator. The forces within the structure were calculated using a truss-based approach, and compared to an allowable bending moment and positive axial force. Furthermore, the motion of the modules was abstracted to a simple sliding square model. These approximations provide a suitable method for validating the principle of the force-aware self-assembly algorithms, and provide a template that can be built upon to deploy the algorithms on physical robots.

In this chapter, the algorithms are implemented on a real-world robotic platform to evaluate how effectively they can perform in a laboratory environment. As explored in Section 2.2, there are a wide range of robotic platforms that the algorithms could possibly be deployed on. HyMod [25] (Figure 2.4d) is chosen for a number of reasons:

- Modules can form square lattice arrangements, emulating the simulated experiments.
- Modules can move around the square lattice by forming a metamodule of two modules and moving with a flipping gait. To do this, they bend at the central axis of each module, referred to as the *body joint*.
- The HiGen connector [26] forms a mechanical connection between agents. Deformations of components that comprise the connector can be measured to calculate a force within the connection in a simpler manner than for other types of connector, such as those relying on electromagnetism.
- The retractable nature of the HiGen connector means modules can rotate in place within the lattice without colliding with their neighbours, reducing the complexity of moving around the lattice.

Despite these promising attributes, the platform required modifications to be made in order to verify these algorithms.

Figure 6.1 illustrates how the cantilever construction algorithm could be implemented with the HyMod platform. A square lattice arrangement of modules extends from a vertical fixed support, which a metamodule of two modules can move along. This metamodule can walk across a horizontal row of other agents on its own using a flipping gait (Figures

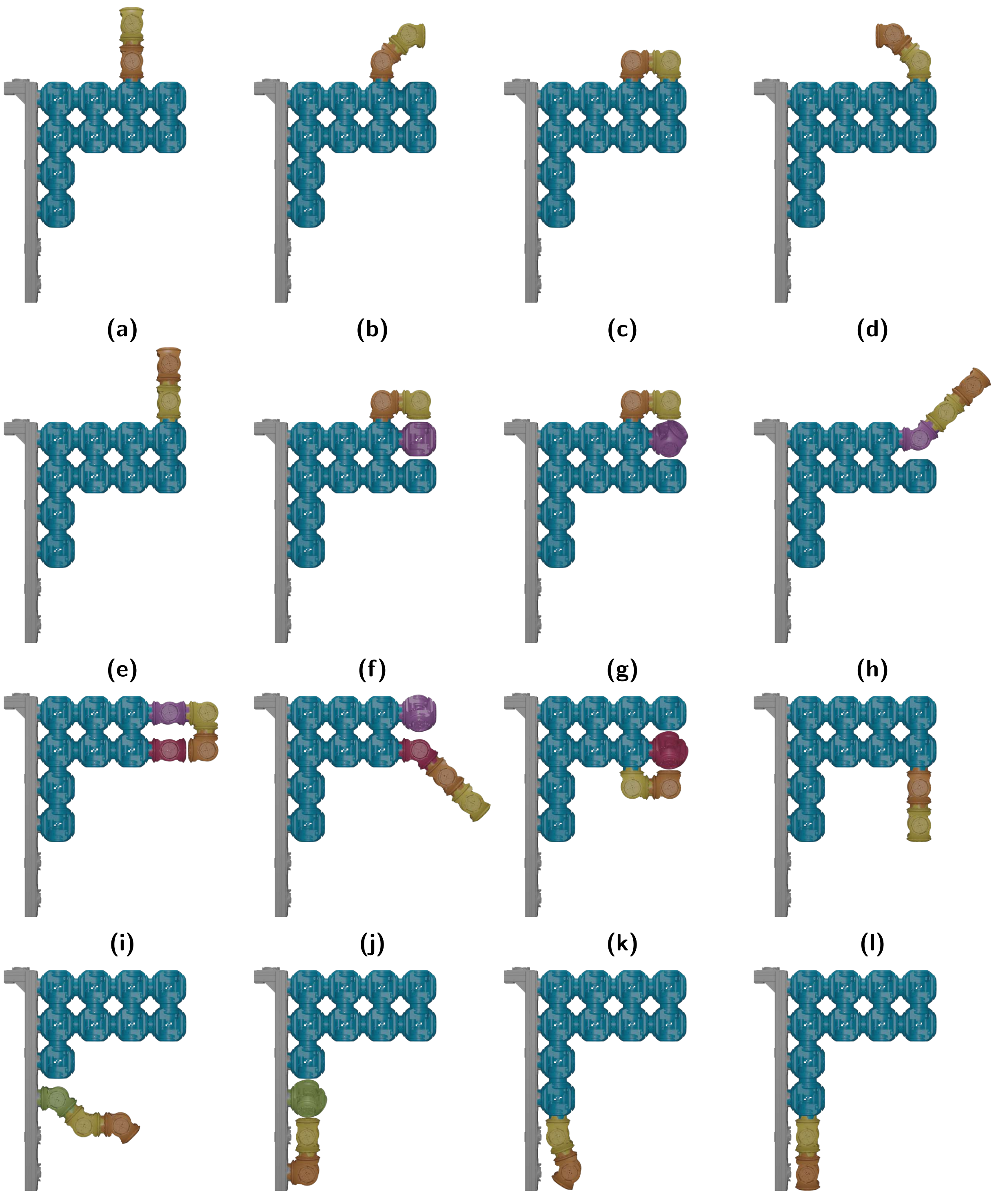


Figure 6.1. Renders showing how an active metamodule (yellow and orange) can move around a structure extending from a fixed support (grey). The structure consists of the blue modules, who occasionally become 'helper' modules (purple, red, and green) to assist the metamodule when passing to a different row. The metamodule moves **(a – e)** one column to the right, **(f – l)** past the tip, and **(m – p)** down a row.

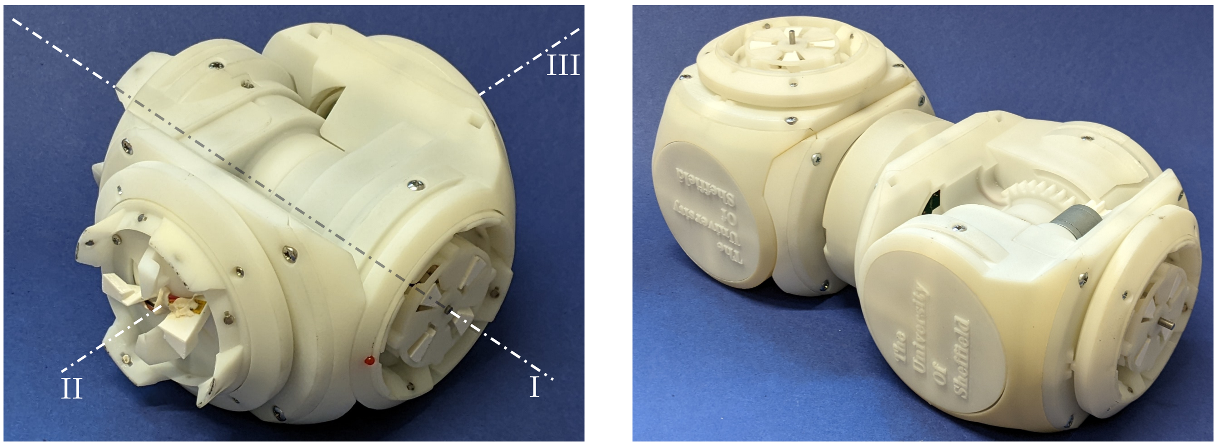


Figure 6.2. **(a)** The new passive HyMod module, and **(b)** the new active metamodule. Each module has the three rotational degrees of freedom shown in **(a)**: a central axis (the body joint, I) and two wheels (II and III)

6.1a – 6.1e). However, it needs assistance from other modules in the structure to pass the tip (Figures 6.1f – 6.1l) or to move to adjacent rows (Figures 6.1m – 6.1p). The design of the HyMod system allows modules to rotate in place without collision when adjacent HiGen connectors are open (Figures 6.1g, 6.1k, and 6.1n). However, modules must tilt away when rotating next to a connector with its docking hooks extended, as is the case for the connectors on the fixed support (Figure 6.1o).

This chapter is organised as follows. The custom HyMod hardware designed for these experiments is described in Section 6.1, and characterised in Section 6.2. In order to reflect the characteristics of the HyMod platform, the self-assembly algorithms had to be slightly modified as described in Section 6.3, before the experiments presented in Section 6.4 could be performed. Finally, the chapter is concluded in Section 6.5.

6.1 Hardware Design

The self-assembly algorithms are implemented using an updated version of the HyMod modular robotic platform. A module from the original version of this platform is shown in Figure 2.4d: it consists of two half shells that can rotate around a central body joint by $\pm 90°$, each of which contains a wheel capable of continuous rotation to allow the robot to move around a flat surface using a differential drive configuration. There are HiGen connectors [26] at each end and on each wheel so the module can connect to others on four faces, therefore creating arbitrary structures that reside in a cubic lattice.

This section describes how, using this design as a starting point, new modules were designed and manufactured with which to verify the self-assembly algorithms in real-life. Firstly, variants of the HiGen connector were made that allow for the forces within each connector to be measured (Section 6.1.1). These were included in a *passive module* (Figure 6.2a) with the same geometry and degrees of freedom as the original HyMod module, but without motors for actuation, instead requiring a user to manually rotate the body joint and wheels (Section 6.1.2). Structures built by hand from these modules allow for verifi-

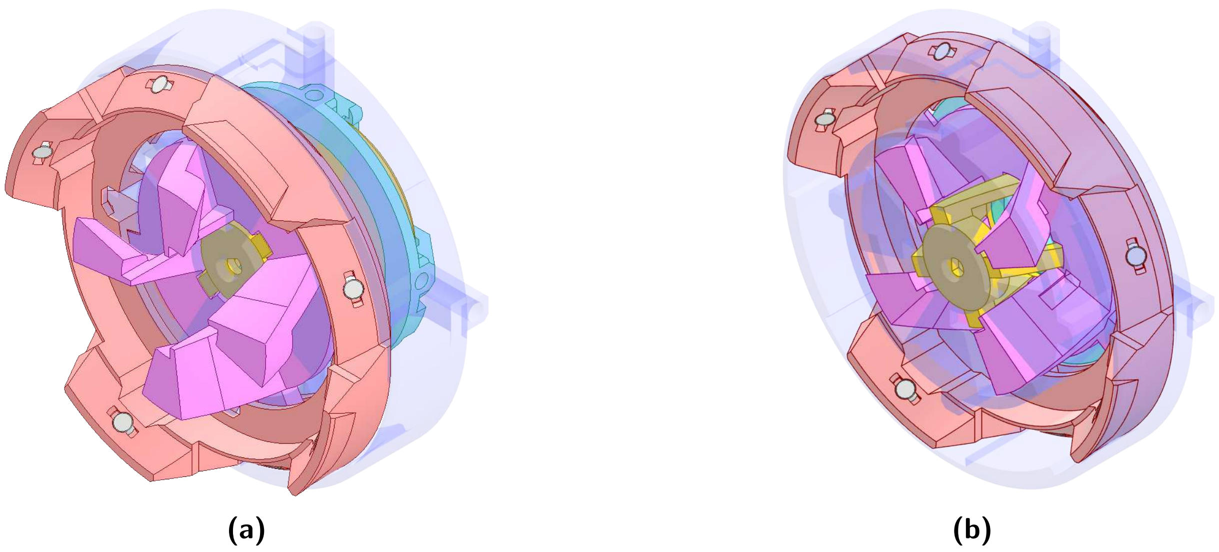


Figure 6.3. CAD render of a HiGen connector, shown both **(a)** closed and **(b)** open. The driveshaft is shown in yellow, the docking hooks in pink, the alignment shroud in peach, the upper outer housing in purple (transparent to reveal the internal components), and the rear outer housing in cyan. Neodymium magnets aid alignment (grey). In sensed connectors, small slots in the docking hooks allow wires to be be connected to strain gauges.

cation of the core concepts of the self-assembly algorithms without being concerned with refining hardware designs to allow for autonomous operation, a considerable undertaking in itself. As a proof-of-concept for how the modules would be able to autonomously move around the lattice structure, a single *active metamodule* (Figure 6.2b) was constructed, consisting of two modules permanently attached in the middle and with fully-actuated body joints in each module (Section 6.1.3).

6.1.1 Force-aware HiGen Connectors

The HiGen connector is a method of forming connections between self-assembling modular robots designed by Parrott *et al.* [26] (Figure 6.3). A small DC motor with a 298:1 gear ratio rotates a central driveshaft (yellow), which rotates four docking hooks (pink). These turn within an alignment shroud (peach), causing both the docking hooks and alignment shroud to extend out of an outer housing (purple, transparent). Once extended, the docking hooks continue to rotate to engage with the hooks of the opposite connector. All the mechanical components save the motor are made from 3D printed plastic. This design has a number of attractive features:

- Connections can be made between any two connectors in discrete 90° intervals, without concern for the gender of opposing connector. Furthermore, each HiGen connector in a connected pair can actuate independently of the other one, meaning if one unit develops a fault the other is still able to disconnect. The connector is therefore genderless, as defined in Section 2.2.2.

- The physical coupling is fast, taking less than 0.3 s.

- The mechanical nature of the connection means that energy is only consumed during actuation, unlike other methods which require energy to be consumed continuously while connections are active, such as electromagnets.
- The extending actuation means that modules such as HyMod that utilise HiGen connectors can occupy less space in a lattice arrangement when the connectors are retracted, hence avoiding collisions while rotating within the lattice.
- The shape of the alignment shroud allows for automatic correction of small misalignments between connectors, which is also aided by the inclusion of small neodymium magnets in the alignment shroud.

The original HiGen connector also includes a PCB on the alignment shroud with spring-loaded pins to allow electrical connections between units, enabling communications and power sharing across the connection. These are not required by the updated HyMod units, and so are excluded for simplicity. Each connector has a PCB mounted to its rear, containing the control electronics and two microswitches that are activated by the driveshaft when at the limits of its motion to inform the controller that the actuation is complete.

A key component of the self-assembly algorithms is that they require measurements of the force within the connections between agents. Past robotic platforms have also incorporated force measurements, such as the CHOBIE II system [67], and the CoBOLD connector used by CoSMO [102], which incorporated strain gauges and force-sensitive resistors respectively. Another design by Melenbrink *et al.* uses low-cost transducers made from Velostat to measure the forces between robotic modules [17].

To obtain measurements of the forces within the HiGen connectors, strain gauges were applied to the docking hooks, as they are a reliable method of sensing force commonly used in engineering applications. However, the readings are highly subjective to proper installation and careful electronic design, so they are relatively expensive to use. For this reason, *finite element analysis* (FEA) using Autodesk Inventor Nastran was performed to determine the minimum number of strain gauges required to obtain enough data to accurately inform the self-assembly algorithms, and the location at which they should be placed. In order to focus the analysis on the deformation of the docking hooks, the basic geometry of a HiGen connector was modelled as a single part instead of the separate components, and the base of the connector was thickened to 13 mm. The connectors were meshed with a global spacing of 1.5 mm, but a finer mesh spacing of 1 mm was used for the docking hooks to obtain more information about this region. Each connector was modelled as acrylonitrile butadiene styrene (ABS), a polymer commonly used in 3D printing, using built-in parameters. Two loading scenarios were considered: an axial tension of 7.95 N, and a combined moment of 0.556 N m and shear force of 7.95 N. These represent the loads associated with supporting an original HyMod module of quoted mass 810 g and lattice spacing 140 mm [25] either vertically or as a horizontal cantilever respectively.

Results of the FEA are shown in Figure 6.4. It can be seen that the positive axial load is shared approximately evenly between the four hooks, which are all loaded in tension. When the weight of a simulated cantilevering module is applied to the connector, the majority of the load is shared between the two hooks in the plane of the rotation, but with a significant load also taken by the alignment shroud. In both cases, the maximum deformation occurs near the base of the docking hooks.

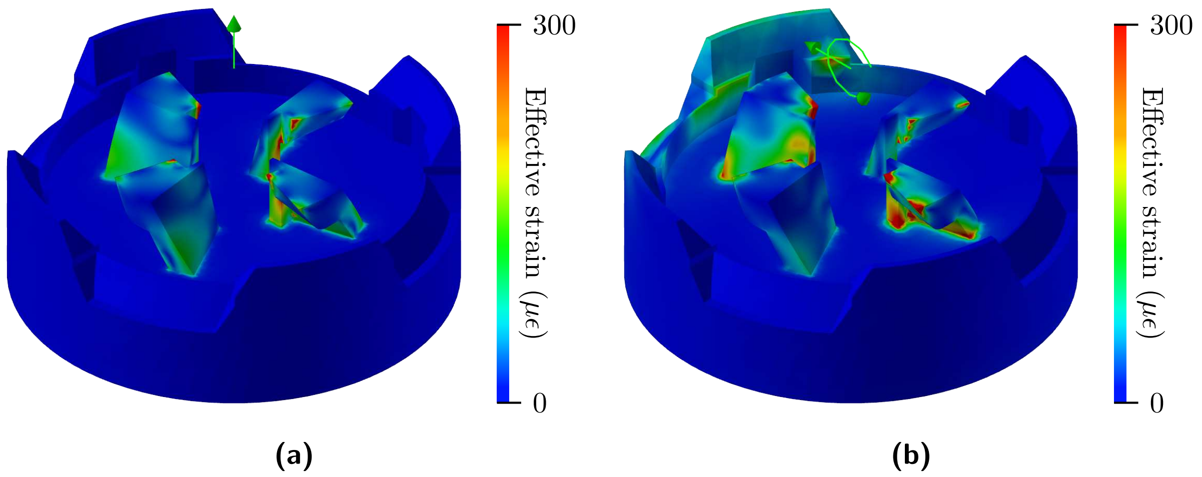

300
Effective strain ($\mu\epsilon$)
0
300
Effective strain ($\mu\epsilon$)
0
(a)
(b)

Figure 6.4. Finite-element analysis of a HiGen connector under **(a)** a positive axial force of 7.95 N, simulating a hanging HyMod module, and **(b)** a combined moment of 0.556 Nm and shear force of 7.95 N, simulating a cantilevering HyMod module. The applied loading is shown by the green arrows.

From these results, it was concluded that two strain gauges are sufficient to obtain enough data to inform the self-assembly algorithms. These are placed on the docking hooks that will be in the plane of rotation when modules are arranged as a cantilever. The gauges are positioned at the base of the outside of the docking hooks so that they measure the largest deformations while not obstructing the mechanical interface between opposing hooks.

In a fully-autonomous system, every HiGen connector in each HyMod module would be capable of actuation, termed *active*, and sensing force, termed *sensed*. However, for simplicity, four different kinds of connector were designed for this work. In addition to the active connectors, ones that were permanently extended, called *passive*, were designed, as were ones without the strain gauges, called *unsensed*. There are therefore four different variants of HiGen connectors:

1. Active sensed (Figure 6.5a)
2. Active unsensed (Figure 6.5b)
3. Passive sensed (Figure 6.5c)
4. Passive unsensed (Figure 6.5d)

The strain gauges selected were the LY91-1.5/350 model from Hotttinger, Brüel & Kjær. These were chosen for their small size and the location of the solder pads next to the sensing direction, instead of in line with the sensing direction, as is more common. Each gauge was bonded to its docking hook using Z70 cyanoacrylate adhesive, then the wires were soldered in place. A small amount of X60 two-component cold curing adhesive was applied on top of the gauge and solder joints to provide strain relief and to protect the sensitive strain gauge from wear occurring as components slide over each other during actuation.

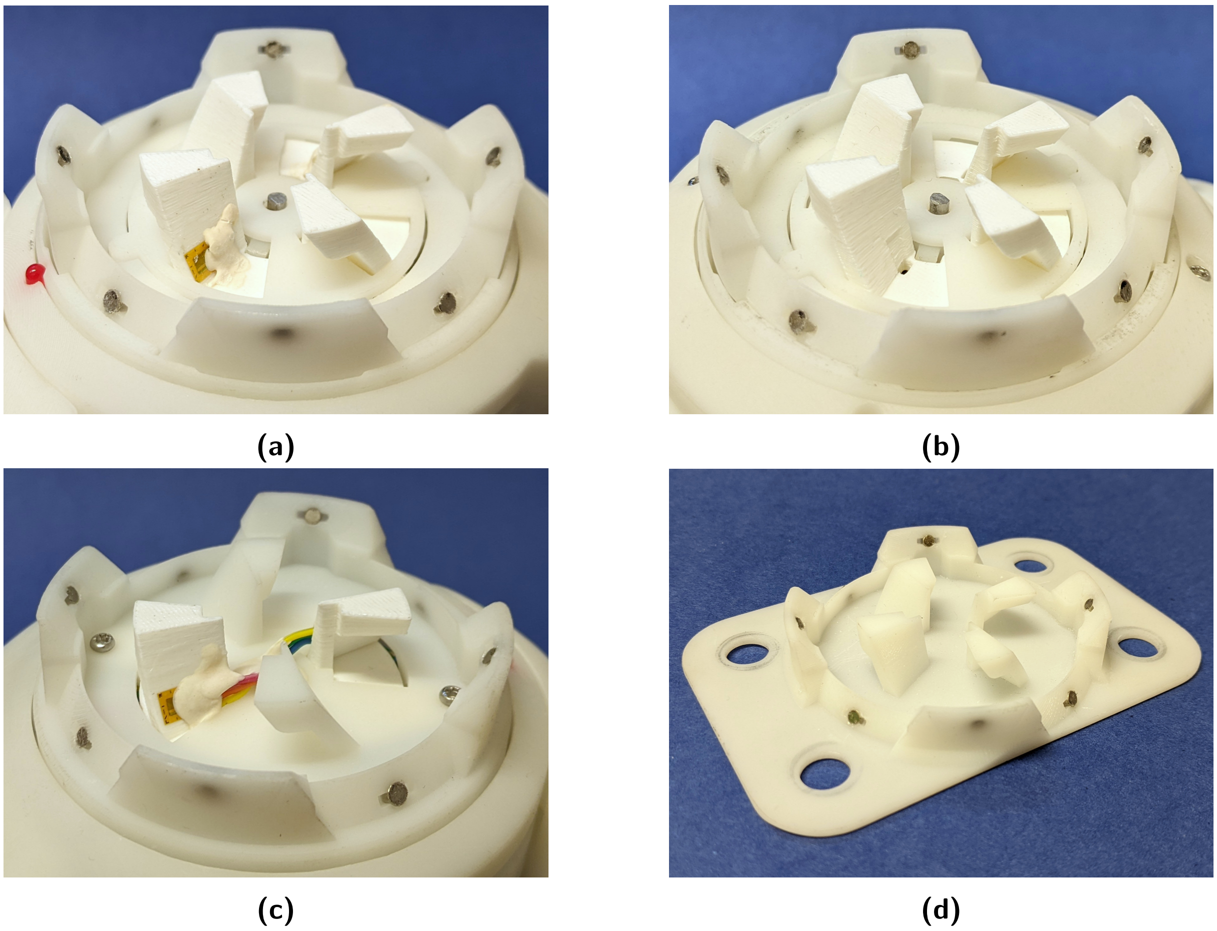


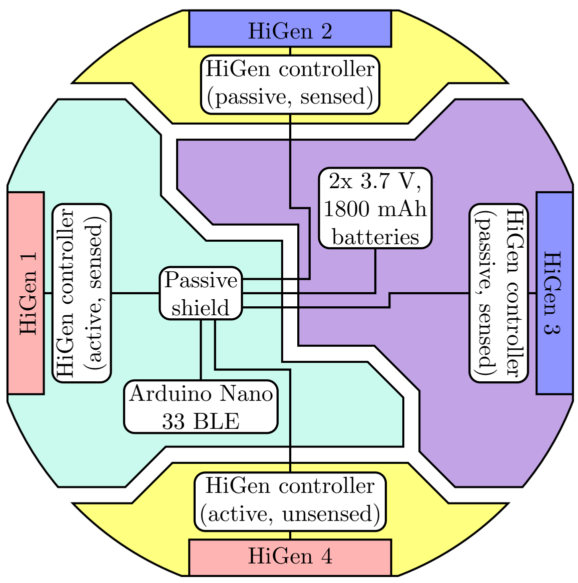
Figure 6.5. The four types of HiGen connector: **(a)** active sensed, **(b)** active unsensed, **(c)** passive sensed, and **(d)** passive unsensed.

The design of the components within the active sensed connectors was modified to ensure that the wires connecting to the strain gauges did not interfere with the mechanism. A small slot and hole was added to each sensed docking hook (visible in Figure 6.3b) as well as to the driveshaft and rear cover of the connector to achieve this.

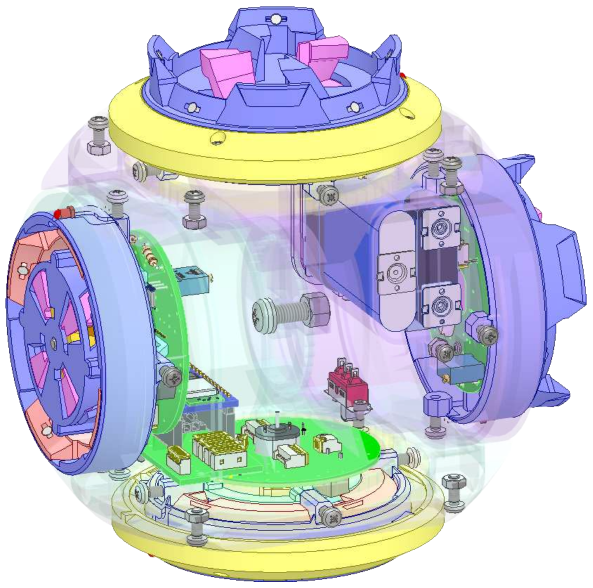
The components, with the exception of the docking hooks, were all 3D printed using a stereolithography (SLA) printing process from Somos LEDO 6060 resin. It was found that Z70 adhesive was unable to successfully bond the strain gauges to the resin parts, so the docking hooks were printed using a fused deposition modelling (FDM) process from ABS instead.

6.1.2 Passive HyMod Modules

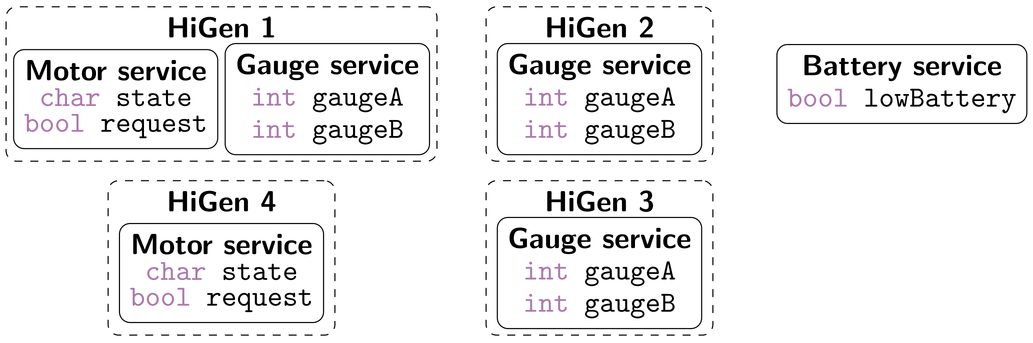
One of the ten passive HyMod modules constructed to verify the self-assembly algorithms is shown in Figure 6.2a. These modules are based on the original HyMod platform [25] (Figure 2.4d), but without motorised actuation in the body joint or wheels: instead, these degrees of freedom can be rotated by hand through discrete 6° intervals. While the original design used active unsensed HiGen connectors in each face, these new modules use a combination of active sensed, active unsensed, and passive sensed connectors to allow


HiGen 2
HiGen controller (passive, sensed)
2x 3.7 V, 1800 mAh batteries
HiGen 1
HiGen controller (active, sensed)
Passive shield
HiGen controller (passive, sensed)
HiGen 3
Arduino Nano 33 BLE
HiGen controller (active, unsensed)
HiGen 4

(a)



(b)



(c)

Figure 6.6. The design of the passive HyMod module. **(a)** The systems diagram. **(b)** A CAD drawing of the complete module, with the external casing transparent to reveal the locations of the internal components, including PCBs and batteries. **(c)** The design of the BLE services and characteristics.

forces to be sensed in at least one connector forming a pair between all modules within a regular lattice arrangement. A systems diagram is shown in Figure 6.6a, including the placement locations of each kind of HiGen, while a CAD model revealing the locations of internal components is shown in Figure 6.6b. All components are 3D printed from Sonos Ledo 6060 resin with the exception of the docking hooks, which are printed from ABS for reasons described in Section 6.1.1.

Each module is powered by two 3.7 V, 1800 mA h lithium-ion batteries connected in series to give a nominal voltage of 7.4 V. These directly power an Arduino Nano 33 BLE (*Bluetooth Low Energy*), mounted on a PCB containing a 5 V linear voltage regulator and headers to allow connections to other PCBS within the module.

The four HiGen connectors each have a dedicated PCB mounted on their rear, with different components depending on the variant of HiGen. The PCBs for active connectors have an H-bridge motor driver which runs the DC motor at the full 7.4 V supply voltage in a direction controlled by two digital pins on the Arduino; a constant voltage applied to a reference pin of the motor driver provides a 67 % duty cycle pulse-width modulated signal to set the motor speed to a reasonable level. These PCBs also feature two microswitches used to determine when the driveshafts are at their clockwise or anti-clockwise limits. The PCBs for sensed connectors contain resistors to connect each strain gauge as one leg of their own Wheatstone bridge. The output of each bridge is connected to an instrumentation amplifier to boost the small voltage differences produced by the gauges as they deform to a level measurable by the built-in analogue-to-digital converter on the Arduino. These amplifiers are powered from the regulated 5 V signal, and the offset for each is tuned during assembly by a rotary potentiometer so that the output in the undeformed state is roughly half of the output saturation voltage. Finally, each HiGen PCB also includes connections to control a light-emitting diode (LED) on the edge of the connector for debugging purposes.

The modules are controlled by a central computer through BLE. A graphical user interface (GUI) was created using Python and Qt6 to allow the motors to be controlled and the measurements from the strain gauges to be viewed. The communication is structured as shown in Figure 6.6c. The motor of each active HiGen is allocated a service with separate characteristics for the requested and actual state of the connector. Each pair of strain gauges in a sensed HiGen is collated into a single service with a characteristic for the reading of each gauge. A further service with a single characteristic informs the user when the 5 V voltage regulator is unable to produce a suitable output voltage, meaning the batteries are so low that the strain gauge readings will be unreliable. In practice, however, the motors will slow down or stop actuating altogether as the batteries drain, prompting the user to recharge the batteries before this warning is issued. In order to simultaneously process HiGen actuation requests and measure the strain gauges, multithreading is implemented using Mbed OS on the Arduino [191] and QThread on the central computer [192].

Each active HiGen is actuated by setting the `request` characteristic of its BLE service. If the requested state is different to the current state, the motor turns in the required direction to open or close the connector, and the `state` characteristic is set to `'a'` to reflect that the motor is active. The motor turns until the corresponding limit switch mounted on the PCB of the connector is pressed, whereupon the motor actively brakes for 100 ms then is switch off. The `state` characteristic is then updated to either `'o'` or `'c'` for the open and closed states respectively. The actuation is expected to take less than 2 s, otherwise it is deemed that there is an obstruction. In this case, if the connector is opening then the motion is stopped, but if it is closing then the direction of actuation is reversed to open the connector so that the user can realign it. When the connector is closed, the limit switches are polled every 10 µs to check the appropriate one is still depressed: if it is not, the motor is briefly actuated to ensure the docking hooks on opposing connectors remain engaged.

The strain gauges on each HiGen are constantly updated. Ten ADC measurements for each gauge are recorded at 10 ms intervals and averaged, with the resulting values written to the corresponding BLE characteristics. These values are used to vary the intensity of the LED adjacent to the connector to provide visual feedback on the current loading, and are converted to a force by the central computer, as described in Section 6.2.2.

6.1.3 Active HyMod Metamodule

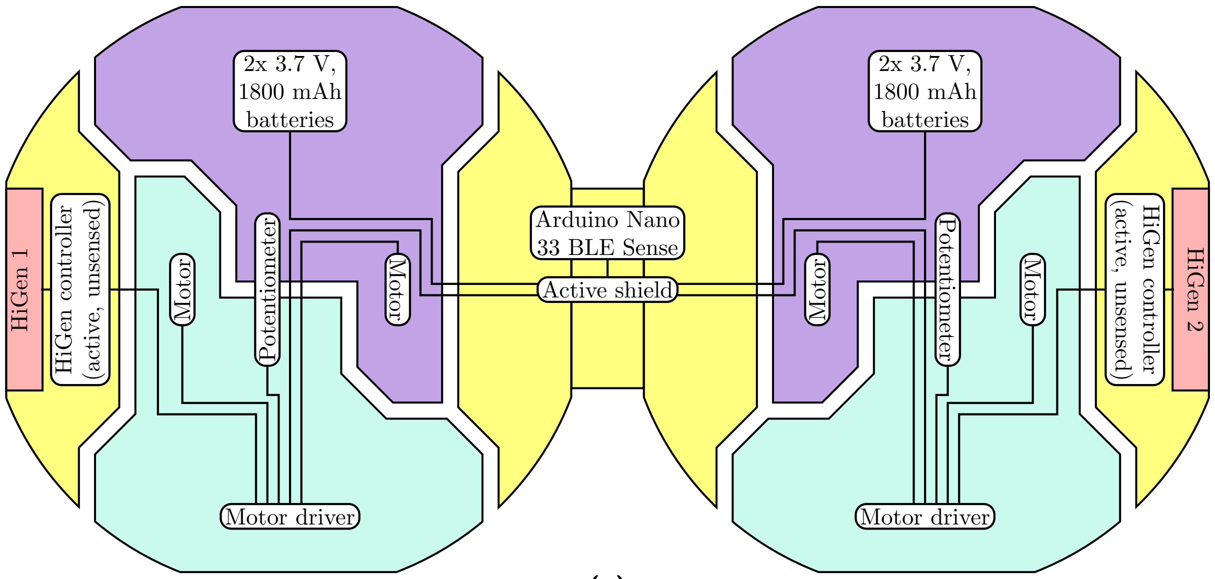
In addition to the ten passive HyMod modules, a single active metamodule was also designed and manufactured, shown in Figure 6.2b. The purpose of this metamodule is to demonstrate how two HyMod modules could cooperate to walk around a lattice of other modules, thus allowing autonomous self-assembly following the force-aware algorithms developed in this thesis.

The active metamodule consists of two identical HyMod modules where the HiGen connectors on one wheel of each module are replaced with a fixed tube to create a permanent link between the two. The opposite wheels both incorporate passive unsensed HiGen connectors, and there are no other connectors on the other faces. A systems diagram is shown in Figure 6.7a, and a CAD model revealing the locations of internal components is shown in Figure 6.7b. Each module contains two of the same 3.7 V, 1800 mA h lithium-ion batteries as in the passive modules, again connected in series to give a nominal supply voltage of 7.4 V. Both pairs are connected in parallel to retain the same supply voltage as a single pair, but allow for a higher capacity and higher current flow. The metamodule is controlled by an Arduino Nano 33 BLE Sense located in the fixed connection between the two modules. Each module contains a PCB that controls the joint motors using full-bridge motor drivers and obtains the measurement from a potentiometer within the body joint as described below. This board also contains headers to connect the Arduino to the HiGen controller PCB belonging to the active unsensed connector on the wheel.

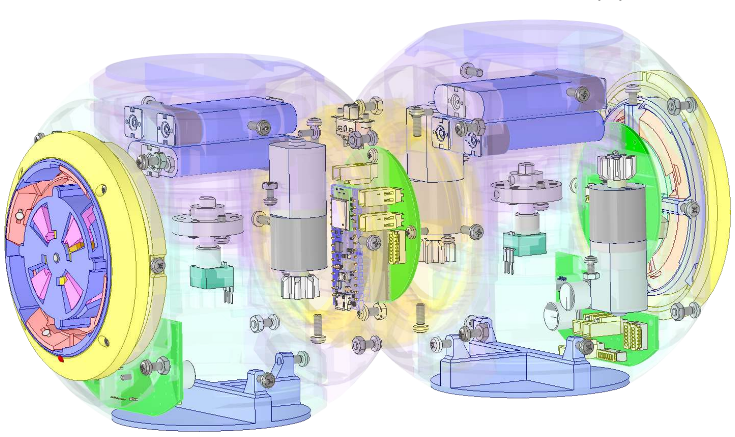
The body joint of the each module is actuated through the same mechanism as the original HyMod modules. Two high-torque DC motors with integrated 154:1 gearboxes work together to turn the joint through its range of motion, with a further 5:1 gear reduction obtained through gears attached to the driveshaft of each motor which engage with others in the body of the module, as shown in Figure 6.7c; the two halves of each module rotate about a central potentiometer, also visible in this figure, that allows the controller to obtain the current joint angle. As in the passive modules, all components except for the ABS docking hooks are printed from Sonos Ledo 6060 resin.

Another Qt6 GUI was created using Python to control the active metamodule through BLE. The BLE communication is structured as shown in Figure 6.7d. The HiGens are unsensed, and actuated through the same service as in the passive modules. The joint is actuated through a separate service. The user sets the `request` characteristic to the angle they would like the joint to move to. The `state` characteristic is set to `'a'` to reflect that the joint is active, and a pulse-width modulated signal is applied to the joint motors. This begins with a low duty cycle, which is steadily increased until the joint is moving at a preset speed, determined by the rate of change in voltage across the joint potentiometer. The duty cycle is constantly updated during motion to ensure the speed is constant, regardless of whether the joint is acting with or against gravity. The motion continues until the specified angle is reached, at which point the motors briefly brake before being turned off. If the potentiometer measurement does not significantly change for 2 s, it is deemed that the joint has stalled and the motors are turned off. When the motors are switched off, the `state` characteristic is set to `'s'` to show the motion has stopped. The `position` characteristic is constantly updated throughout the motion to report the current joint angle.

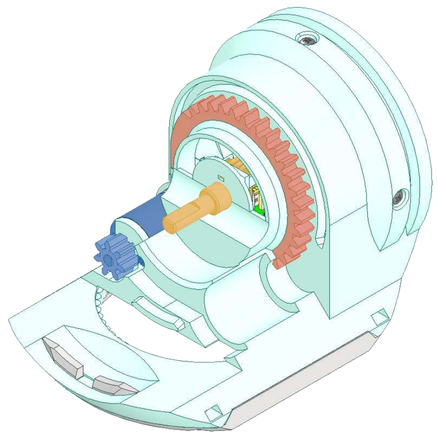
The Arduino Nano 33 BLE Sense features an inertial measurement unit (IMU), which is used to determine the orientation of the metamodule. Since the metamodule is designed



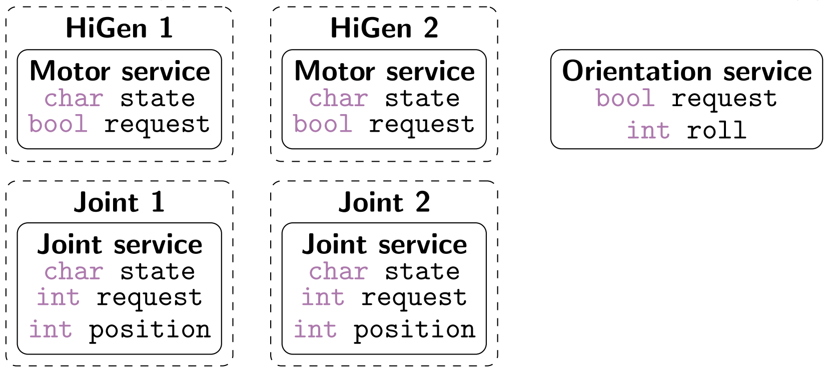
(a)



(b)



(c)



(d)

Figure 6.7. The design of the active HyMod metamodule. **(a)** The systems diagram. **(b)** A CAD drawing of the complete metamodule, with the external casing transparent to reveal the locations of the internal components, including PCBs and batteries. **(c)** A CAD drawing of one half of a module of the active metamodule, showing the motor with an additional gear (blue) that engages with the teeth on the opposite half to provide a 5:1 gear reduction (orange), and the potentiometer that forms the body joint (yellow). **(d)** The design of the BLE services and characteristics.

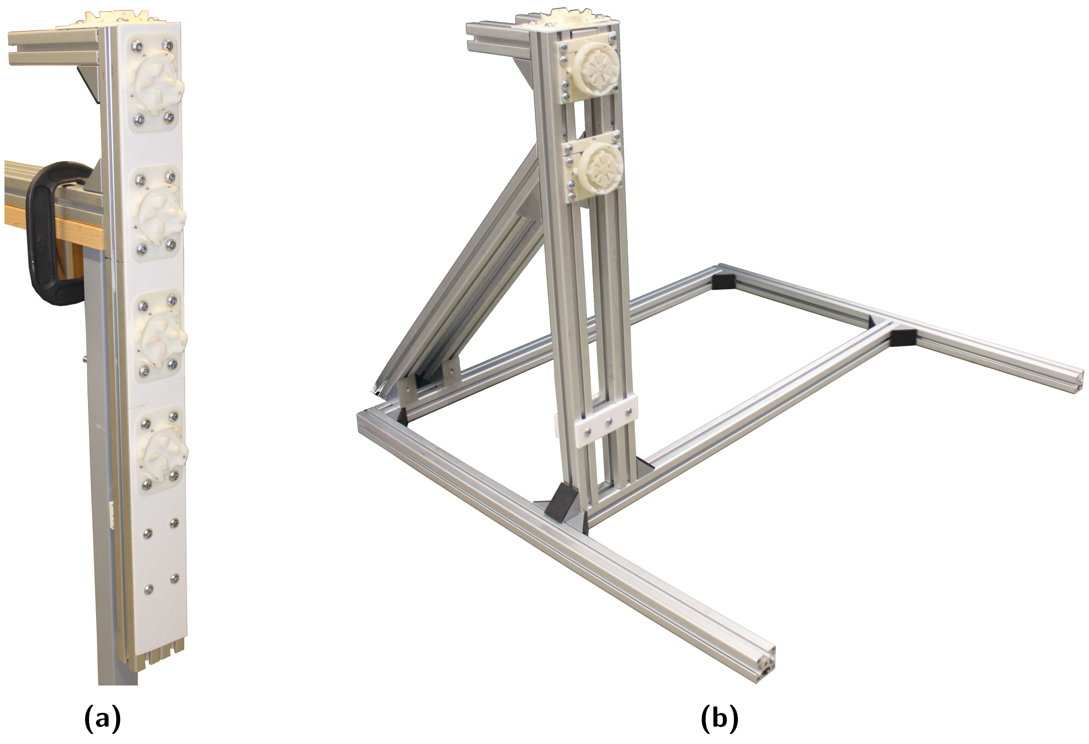


Figure 6.8. The two vertical supports between which the structures were built: **(a)** the left support, clamped to a desk, and **(b)** the freestanding right support.

to be used in two dimensions, this can be described using only the roll angle, reported by the `roll` characteristic in the orientation service. This is calculated using only the accelerometer on the IMU without the complex algorithms required to accurately update 3D orientation, such as the Madgwick filter [193]. This is updated automatically as joints move, and can also be manually updated with the `request` characteristic of the orientation service. The roll angle is used to draw the metamodule on the GUI in the correct orientation, but could also be used in future to assist with accurate module motion.

6.1.4 Vertical Supports

Specialist vertical fixed supports are required between which the HyMod structures are built (Figure 6.8). These were constructed from 40×40 mm aluminium struts due to their strength and versatility. Cantilevers initially extend from the left support, which contains four or five passive unsensed HiGen connectors (Figure 6.5d), depending on the test being performed. These connectors do not have strain gauges bonded to them, so are manufactured from Somos LEDO 6060. This support is clamped to a desk, while the other support is freestanding to allow it to be rapidly repositioned depending on the structure length. The freestanding support contains two active unsensed HiGen connectors with their driveshafts modified such that they can be actuated by hand. One support must utilise active connectors since each pair of opposing faces of the passive HyMod modules contains one active and one passive HiGen connector. Both supports also have a passive unsensed HiGen connector oriented horizontally above the vertical surface.

6.2 Hardware Characterisation

This section describes the tests performed with the hardware to determine its capabilities and limits before implementing the self-assembly algorithms in real-life. The failure strength of the HiGen connectors was first determined (Section 6.2.1), before the strain gauges could be calibrated (Section 6.2.2). Finally, tests were performed to determine how effectively the active metamodule could autonomously walk over surfaces (Section 6.2.3).

6.2.1 HiGen Capacity Testing

The failure strength of the HiGen connectors was measured using a Hounsfield HK100 tensometer set up as shown in Figure 6.9a. An active unsensed HiGen connector was attached to the crosshead with a frame built from 1.5 mm thick mild steel (Figure 6.9b) through a 1 kN load cell, and a second HiGen connector was attached to the machine bed. The lower connector was varied between the active unsensed, passive unsensed, or passive sensed designs, all without the sensing electronics, using the jigs in Figures 6.9c – 6.9e. This allowed for the different combinations of mechanical interface present in structures constructed by these modules to be tested. The load cell readings were recorded as the crosshead was slowly raised until the material failure of one of the connectors occurred. Two trials were performed for each type of connector attached to the machine bed.

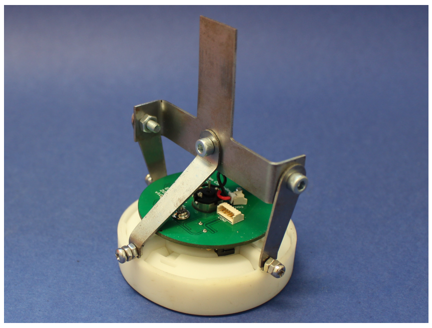
The results for these trials are plotted in Figure 6.10. The mean maximum force measured before failure was 174 N, with a standard deviation of 27.4 N. The FEA in Section 6.1.1 indicates that the purely axial force exerted by the crosshead is spread evenly across each docking hook, so each has a mean capacity of 43.5 N.

In all cases the failure occurred in one or more docking hooks of an active HiGen connector between the 3D printed layers, as shown by the failed docking hooks in Figure 6.11. This indicates that the parts are weaker when 3D printed from ABS using the FDM process than the SLA printed LEDO 6060, and that the weakest point is the links between adjacent layers of ABS. As expected from the FEA modelling (Figure 6.4) the failures occurred near the bottom of the hooks where the highest stresses are predicted.

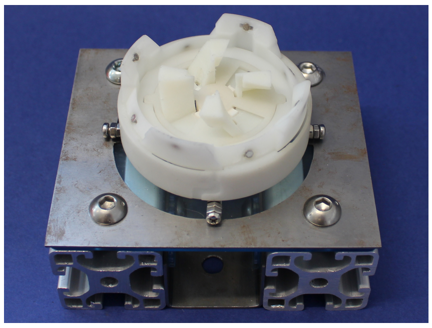
6.2.2 Sensor Calibration

The strain gauges in the sensed HiGen connectors required calibration before they could reliably report the force in their corresponding hook in Newtons. Each gauge was calibrated individually using the same tensometer as for capacity testing, as shown in Figure 6.12a. The passive HyMod module in under test is mounted to the bed of the tensometer using a custom-designed jig shown in Figure 6.12b with the gauge under test oriented on the top. The jig holds the module in place without the use of any of the HiGen connectors to reduce the deformation of the module at locations apart from the connector under test. A modified active unsensed HiGen connector with only one docking hook (Figure 6.12c) was attached through the 1 kN load cell to only apply force to docking hook that is being calibrated.

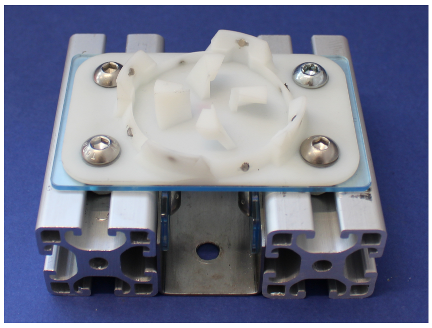
Force was applied to the docking hook under test by slowly raising the crosshead of the tensometer at the minimum rate of 50 mm min^{-1} until the load cell reading reached specified forces, whereupon this force and the measurement in the strain gauge were



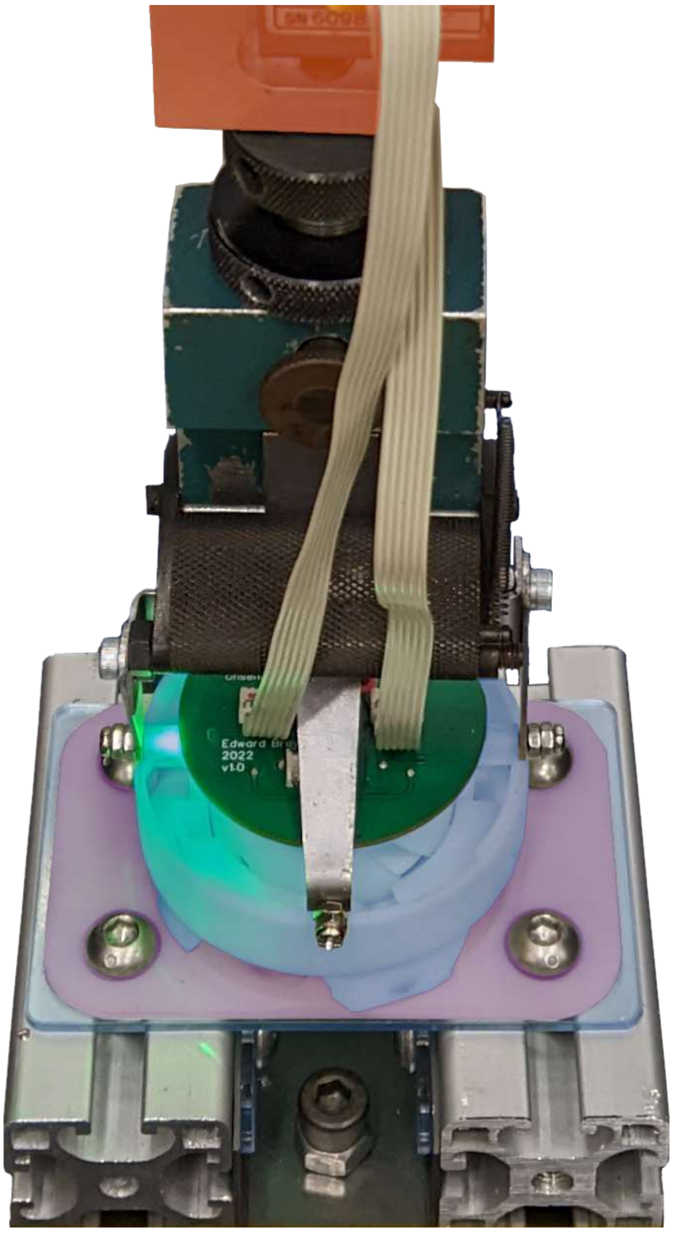
(b)



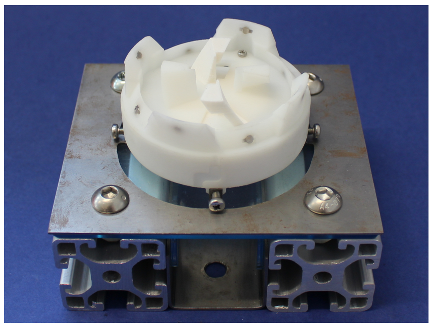
(c)



(d)



(a)



(e)

Figure 6.9. The hardware used during HiGen connector capacity testing. **(a)** An example configuration of the tensometer: an active unsensed HiGen connector (blue) is connected to the crosshead of the tensometer through a 1 kN load cell (orange). The unit under test, in this case a passive unsensed HiGen connector (purple), is attached to the machine bed. **(b)** The active unsensed connector attached to the load cell. **(c – e)** The jigs used to attach the active unsensed, passive unsensed, and passive sensed HiGen connectors respectively to the machine bed.

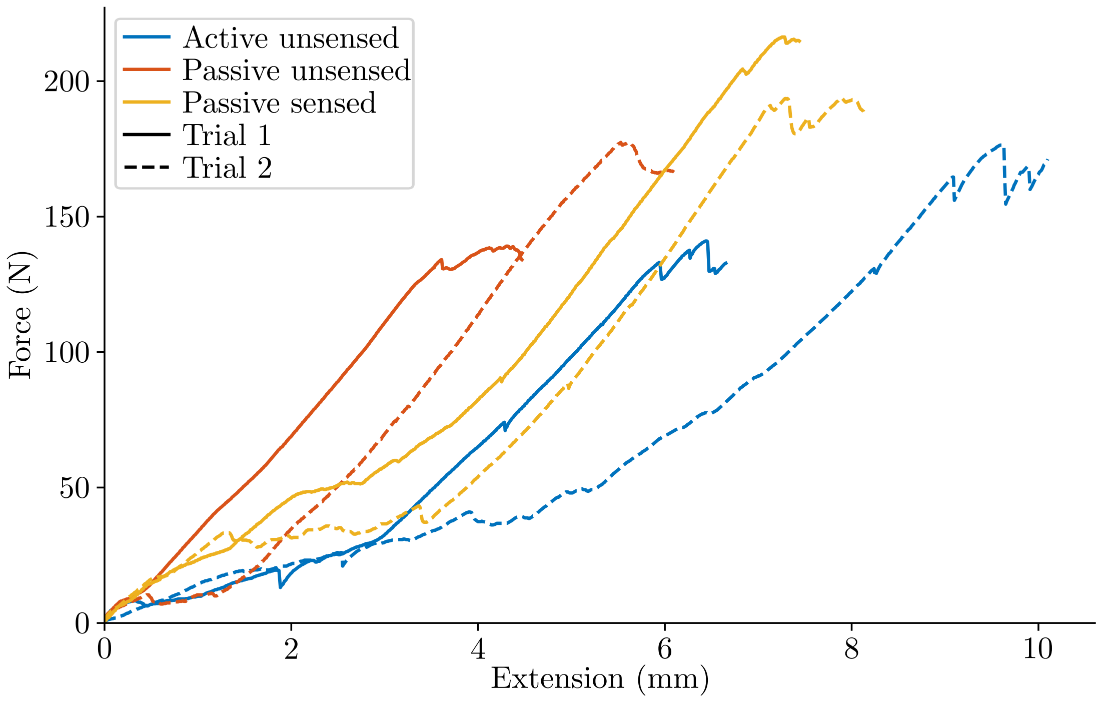


Figure 6.10. The force measured by the load cell during HiGen connector capacity testing as the tensometer pulled two connectors apart. Lines are coloured by the connector on the tensometer bed, and the linestyle indicates the trial number.

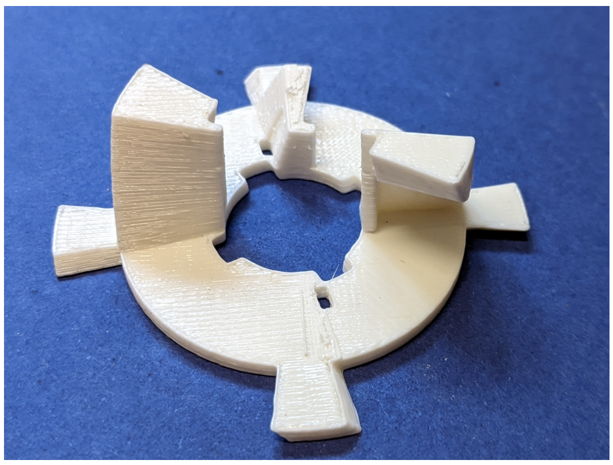


Figure 6.11. An example failure that occurred during capacity testing. This part was used during the second trial with a passive sensed connector on the machine bed.

recorded. The force was varied from 0 N to around 30 N in increments of roughly 2 N to obtain a calibration for each hook up to a value roughly two standard deviations away from the mean hook strength. This range is sufficiently wide for the self-assembly algorithms to be implemented without risking breaking any hooks during calibration. This process was repeated for each of the six sensed hooks on each of the ten passive modules to obtain calibration graphs similar to those in Figure 6.13. A straight line is fitted to each set of points to convert strain gauge readings into forces within each docking hook.

It should be noted that this calibration will not be perfect due to real-world factors such as backlash within the connector changing the exact loading on each hook. However this process gives reasonable calibrations, and effectively means that the self-assembly algorithms are implemented with a moderate amount of sensor noise.

6.2.3 Active Metamodule Motions

The design of the HyMod platform allows a metamodule of two modules to walk around a lattice of other modules, as shown in Figure 6.1. Experiments were performed to evaluate how well the active metamodule could demonstrate this behaviour in real-life. Two scenarios were considered to show the active metamodule moving in a variety of orientations: walking across the top of a bridge of passive modules between the two fixed supports, and moving along the bottom surface of this bridge. Experiments to test passing the tip or changing row (Figures 6.1f – 6.1p) were not performed, as they require one half of the active metamodule to lift two other modules. Work with the previous HyMod platform found that a single module was capable of lifting 1.8 other modules in-line before the 3D printed gear in the body joint and the docking hooks failed [57]. While the stated torques of the motors used in this joint suggest lifting two other modules would be possible if these components were made stronger, it was chosen not to test this capacity to avoid damaging the active metamodule.

6.2.3.A Above a Bridge

The first experiment considered how the active metamodule could walk across the top of a bridge. Two passive HyMod modules were connected in a row between the two fixed supports. The active metamodule was connected vertically above the left support, as shown in Figure 6.14a. The active metamodule was controlled to walk rightwards across the bridge by manually specifying angles for the body joint in each half to be turned to. The HiGen connectors were actuated remotely when required, and this actuation was repeated until successful connections were made. Eleven trials were performed immediately after each other, each ending when the active metamodule was in a vertical position above the right support. The batteries were fully charged before the first trial.

Figure 6.14 shows frames from one trial, and is representative of the behaviour observed across all trials. First, note from Figure 6.14a that the initial position is not exactly vertical: the backlash in the body joint prevents accurate body joint angles from being achieved using only feedback from the potentiometers in each body joint. From the initial position, the active metamodule could easily rotate into a horizontal position (Figures 6.14a – 6.14d). However the alignment between the two HiGen connectors that were brought into contact by this motion was greater than could be corrected for automatically

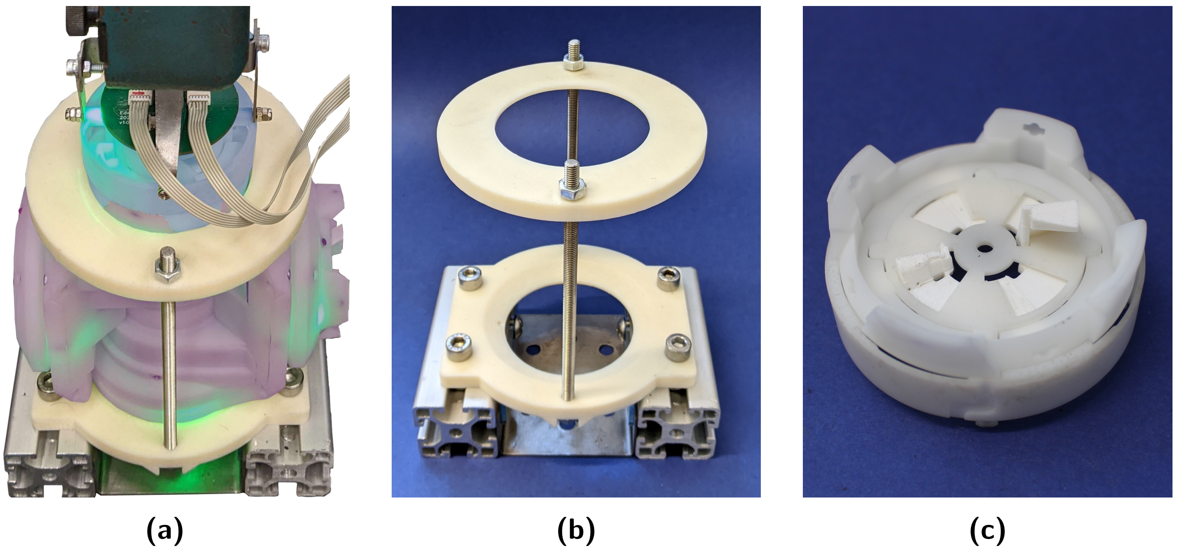


Figure 6.12. The hardware used during strain gauge calibration. **(a)** The configuration of the tensometer: a passive HyMod module (purple) is attached to the tensometer bed with the HiGen under test facing upwards, and an active unsensed HiGen connector with only one docking hook (blue) is connected to the crosshead of the tensometer through a 1 kN load cell. **(b)** The custom jig used to retain the HyMod module in the tensometer. **(c)** The active unsensed HiGen connector with a single docking hook.

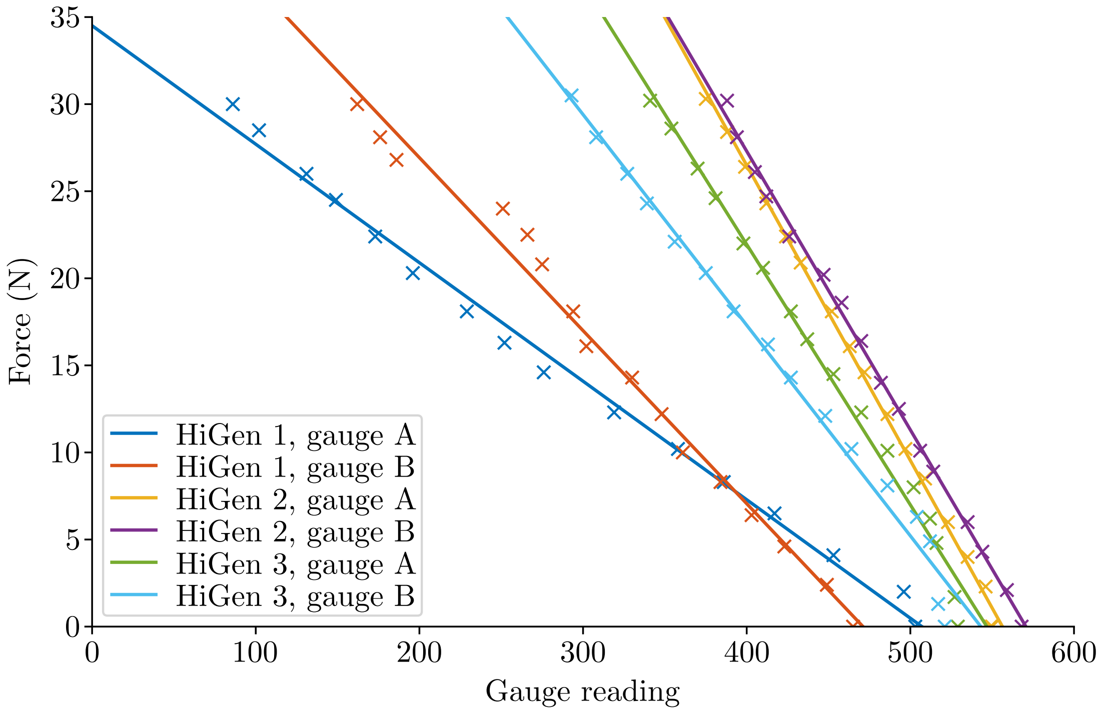


Figure 6.13. Example strain gauge calibrations for one of the passive HyMod modules.

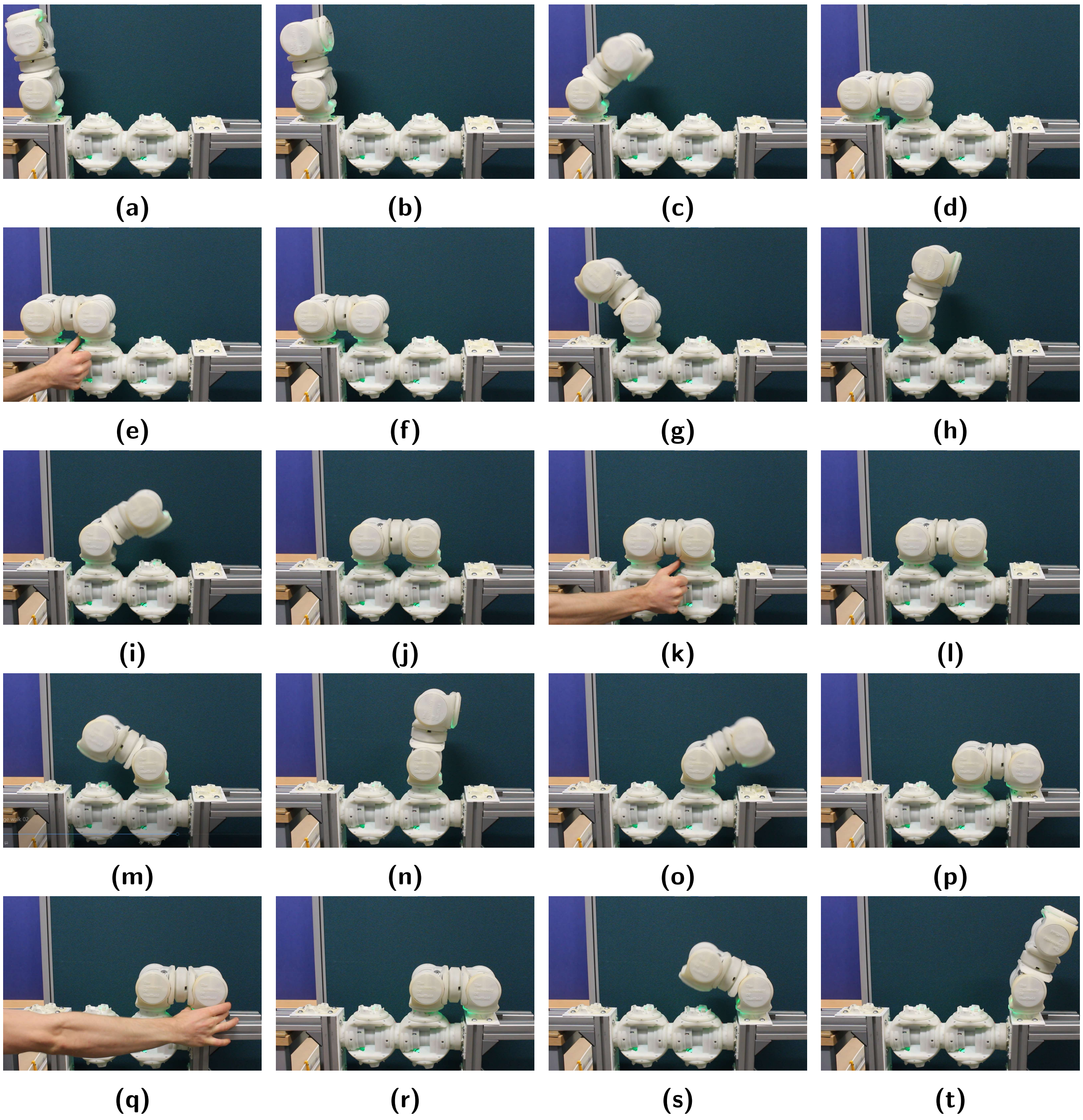


Figure 6.14. Frames from a trial showing the active metamodule walking along the top of a bridge.

by the alignment shrouds due to backlash within the body joints. It was therefore necessary to manually push the connectors into alignment, as shown in Figure 6.14e, before the connection could be made and the HiGen at the other end of the active metamodule released (Figure 6.14f). The metamodule then rotated to a vertical orientation successfully (Figures 6.14g – 6.14h), and the process repeated until the metamodule reached a vertical orientation above the right support (Figures 6.14i – 6.14t).

The results show a mixed performance. On the one hand, the body joint could always lift the full weight of the metamodule and actuate it through the full range of motion, even after the batteries had drained from the previous trials. Actuation was also fast: rotating the body joint through its full range of motion took roughly 2 s when unloaded, and 3 s when lifting the other half of the metamodule. On the other hand, backlash within the body joints prevented autonomous operation and accurate positioning from being achieved. This could be improved by manufacturing the components from a stronger material and to tighter tolerances to reduce the clearance required between moving parts.

6.2.3.B Below a Bridge

The second experiment considered how the active metamodule could walk across the bottom of a bridge. The same bridge as the previous experiment was used, consisting of two passive modules in a line between the vertical fixed supports. Trials began with the active metamodule connected to the passive HiGen connector on the left support in row 3, with the body joint by this connector set to a 90° angle such that the metamodule pointed towards the floor, as shown in Figure 6.15a. The metamodule was teleoperated by a human in the same manner as the previous experiment. It was instructed to rotate to face the leftmost passive module, connect to it, then move to a vertical orientation below the other passive module. The experiment took place immediately after the previous one, so the batteries were partially discharged. Six trials were performed, and the batteries and were not recharged between trials.

Frames from a representative trial are shown in Figure 6.15. The active metamodule successfully rotates to face the leftmost passive module, but there is a large gap between the opposing HiGen connectors of these modules (Figures 6.15a – 6.15d). This gap is due to the backlash in the body joint discussed above, but is also magnified by backlash in the HiGen connector supporting the active metamodule. This has to be compensated for by human intervention (Figure 6.15e). The active metamodule swaps which of its HiGen connectors is supporting it, then rotates to face the next passive module (Figures 6.15f – 6.15j). Once again, human intervention is required to connect to the next module, shown in Figure 6.15k. The active metamodule then moves to a downward-facing orientation, and the trial finishes (Figures 6.15l – 6.15o).

As in the first experiment, the results from these six trials demonstrate the sufficient strength of the motors within the body joint, but highlight issues with backlash. The first rotation performed in these trials is the most challenging tested, as it requires one body joint to lift both halves of the metamodule through 180° against gravity. This was always successful, demonstrating the power in the motors and drive circuitry. However, the backlash issues are more pronounced here, as gravity reveals the HiGen connectors move quite significantly when connected. Similar to the body joints, backlash within the HiGen connectors could be reduced by revising the designs to reduce the clearance between the moving parts, and manufacturing to tighter tolerances.

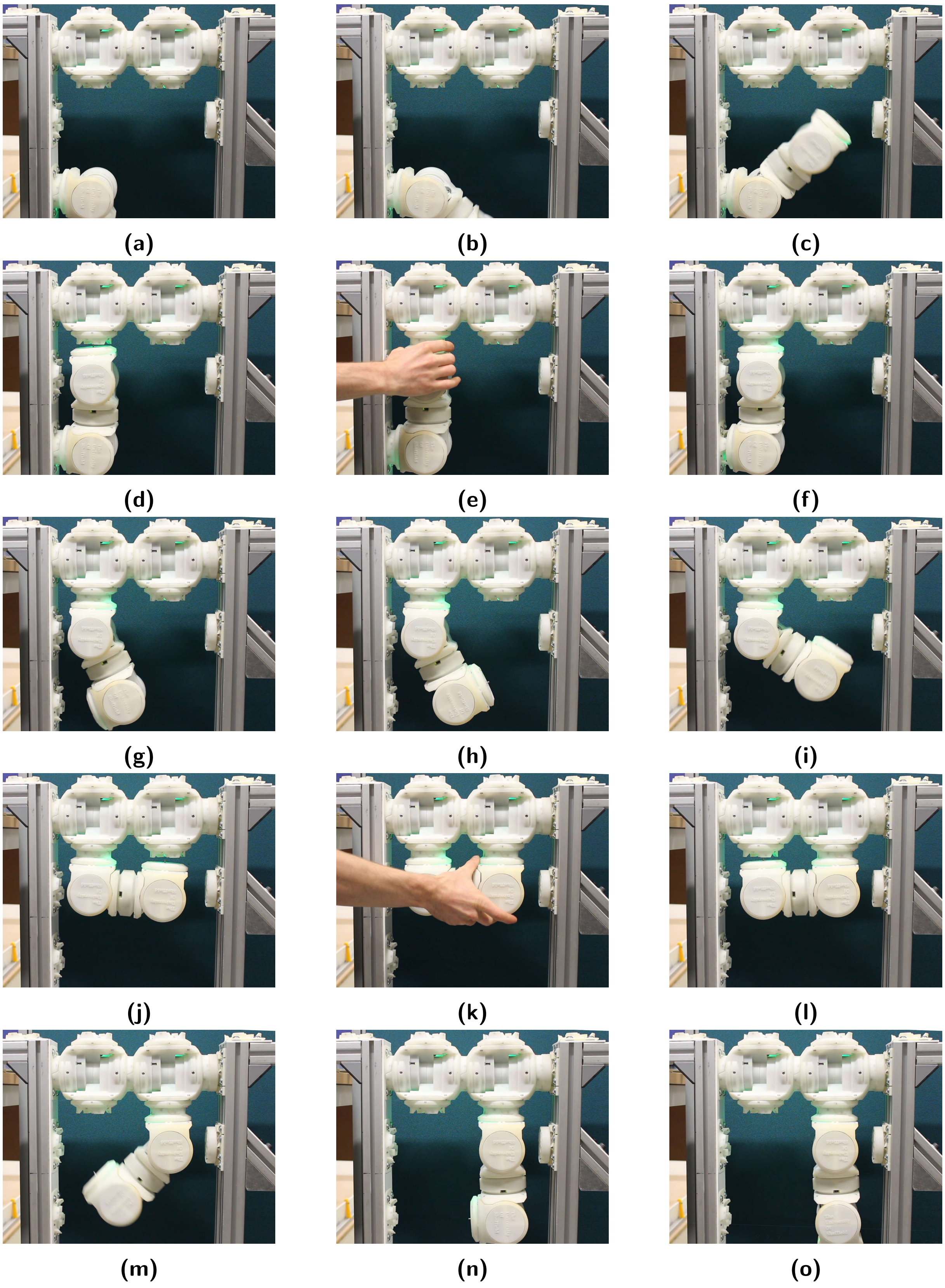


Figure 6.15. Frames from a trial showing the active metamodule walking along the bottom of a bridge.

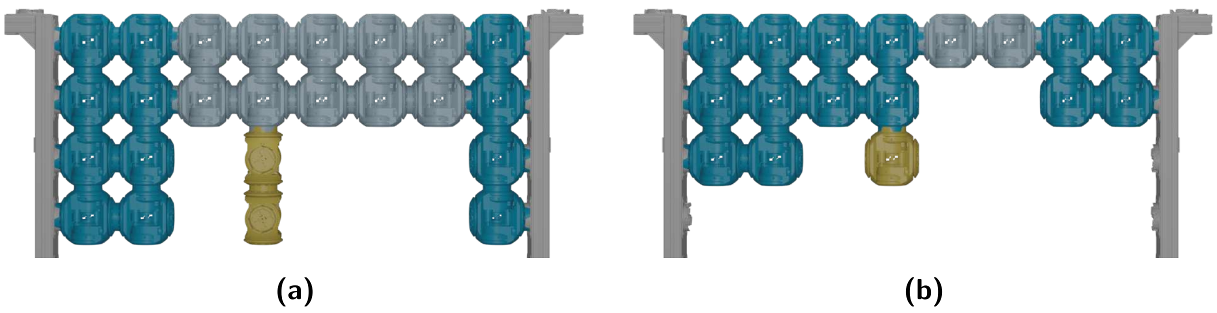


Figure 6.16. The environment considered in this chapter. The HyMod modules in the structure (blue) arrange themselves between one or two fixed supports (grey). The active agent (yellow) moves around the structure to decide its course of action. Two scenarios are considered based on how the active agent is realised: **(a)** as a metamodule agent consisting of two modules oriented vertically above each other, and **(b)** as a single module. Both active agents are shown in position $\{3, 4\}$. The narrow section is shown paler.

6.3 Algorithm Adaptions

The simulator used to develop the self-assembly algorithms in the previous chapters models the agents as sliding squares, and calculates an equivalent moment M and axial force F in links between agents. This is slightly different to the capabilities of the developed HyMod system, so the algorithms were updated to reflect this. They were then implemented using the constructed HyMod modules, which were moved around the structure by hand to validate the core concepts of the algorithms. The situation considered is broadly the same as for the simulated agents, with the differences explained below.

The physical implementation of the self-assembly algorithms presented here considers a group of HyMod robots that can occupy locations within a 2D grid. Positions within this grid are referred to in $(row, column)$ format as in previous chapters.

Individual robots are referred to as *modules*. The morphology of the HyMod platform requires modules to move around the grid as a metamodule of two modules, so two scenarios are considered: one where modules can move on their own, and another where they move as such a metamodule. The former scenario is less realistic, but allows a wider range of structures to be built with the limited number of modules manufactured and thus to validate the algorithms to a greater degree. The term *agent* is therefore used to mean either a metamodule or a single module, depending on the scenario. Agents can either be placed or active as described in previous chapters. Two categories of structure are constructed: those built by *an active metamodule* or by *a single active module*. Placed metamodules could be oriented vertically or horizontally: the vertical orientation is chosen here to build shorter and stronger structures than would be built from the horizontal orientation. Examples of this grid for structures built by an active metamodule and a single active module are shown in Figures 6.16a & 6.16b respectively.

Positions of agents within the grid are given in braces, where $\{r, c\}$ refers either to a single module in (r, c), or a metamodule occupying both (r, c) and either $(r+1, c)$ if $r > 0$ or $(r-1, c)$ if $r \leq 0$. The fixed supports are placed within the grid in the manner described in earlier chapters. The structure is again restricted to be continuous, and structures with a fixed support at both ends are once more required to have a single narrow section: this is a single module thick for structures built by a single active module, but two modules thick if an active metamodule is used, as shown in Figure 6.16.

The sensed HiGen connectors are capable of measuring a force in two of their docking hooks. The FEA indicates that the modules could convert this into a separate moment and axial force, as an axial force will cause both hooks to extend, while a moment will cause one to extend and the other to compress (Figure 6.4). However, in practice it was found that calibrating the strain gauges to accurately report these measurements separately was not possible due to backlash in the connectors. Instead, it was chosen to use a single measurement H to represent how close a connector is to failing, taken as the maximum force measured in either strain gauge of this connector. This is compared to an allowable value $H_{allowable}$ to calculate the criticalness γ. The previous equation for γ (3.4) is therefore changed to:

$$\gamma = \frac{H}{H_{allowable}} \tag{6.1}$$

H is positive when connectors are pulled apart from each other, so this definition assumes connectors are sufficiently strong in compression that high forces of this type can be ignored. This is reasonable as compressive forces will be shared along the whole contact area of the two connecting alignment shrouds. Note that a link consists of two HiGen connectors. It is therefore possible that the value of γ in a link could be reported differently by each of the constituent modules depending on differences in their calibration or other physical factors.

Due to the limited number of modules built, trials were only conducted with one active agent at a time. The sequential cantilever construction algorithm was therefore implemented, and the bridge optimisation and deconstruction algorithms were modified to also implement them in a sequential manner. The collision avoidance behaviours incorporated by the parallel algorithms when multiple agents intend to move into the same location are therefore not implemented here. Active agents only attach to the structure through a single active link, but the rules for choosing it are simplified in this single agent case: it is set to the link on the top or bottom of the agent depending on if it is below or above row 1 respectively. The specific modifications made to each algorithm to implement them in this real-life scenario are described below.

6.3.1 Cantilever Construction

The cantilever construction algorithm begins with a placed agent in position $\{1, 1\}$. The local variant of the sequential algorithm is implemented with minimal modifications, as shown in Algorithm 6.1. The only differences stem from a single value H being recorded by the active agent from each placed module instead of collecting separate values of M and F (Line 4). These values are recorded in two arrays $\{\boldsymbol{\gamma}^{\beta}\ \forall\ \beta \in \{row, column\}\}$ where four arrays were previously required. The values of H communicated are those recorded by the modules on the perimeter in their connectors in row and column links. The original version of the algorithm takes the maximum measurements in links on the top and bottom of the module for $\beta = column$, but the passive HyMod modules manufactured can only measure forces in hooks in their bottom connector so this value is always used[1].

[1]Note that when the active agent is on the bottom of the structure, this measurement will be the weight of the active agent and thus not related to the force distribution throughout the structure. However, this

Algorithm 6.1. The cantilever construction self-assembly algorithm implemented on the HyMod platform, emphasising differences with the local variant of the sequential self-assembly algorithm implemented in simulation (Algorithm 3.2).

```
1  Initialise at $\{0, 0\}$;
2  while not ($row > 0$ and $column = 1$) do
       // Gathering mode
3      Make one step clockwise;
4      Record $H$ from module below or above into $\gamma_c^\beta$;
5  if No link is critical then
       // Placing mode, to extend
6      Place at tip;
7  else
       // Placing mode, to reinforce
8      Calculate $p_{column}(c)$ from $\boldsymbol{\gamma}^\beta$;
9      while Not placed do
10         $c_{target} \leftarrow$ sample from $p_{column}(c)$ without replacement;
11         ~~Move to column $c_{target}$;~~
12         if Valid location then
13             Place in column $c_{target}$;
```

Calculation of $p_{column}(c)$ in Line 8 is slightly different due to the modified data received by the active agent. Urgency distributions are calculated in the same manner as before, but they are only calculated for H, not M and F. The distributions are therefore denoted $\boldsymbol{\nu}_c^\beta$, and the x^{th} component of $\boldsymbol{\nu}_c^\beta$ is thus given by:

$$\boldsymbol{\nu}_c^\beta|_x = (\gamma_c^\beta)^2 \cdot \frac{\gamma_c^\beta}{\sqrt{2\pi}} \exp\left(-\frac{(\gamma_c^\beta)^2(x-c)^2}{2}\right) \tag{6.2}$$

There are now only $2L$ urgency distributions for a cantilever of length L, which are summed to calculate the combined urgency distribution $\hat{\boldsymbol{\nu}}$ as:

$$\hat{\boldsymbol{\nu}} = \sum_{\beta \in \{row, column\}} \sum_{c=1}^{L} \boldsymbol{\nu}_c^\beta \tag{6.3}$$

This is converted into $p_{column}(c)$ as before.

The active agent decides where it should place following this procedure. Samples are drawn from $p_{column}(c)$ without replacement until a valid column is selected. Since there is only one active agent at a time, this active agent can remember which columns are valid and thus can place directly in the first valid column drawn as opposed to being required to move to each column in turn (Line 13). Instead of placing this physical robot in that location, one or two other passive HyMod modules are placed here, depending on whether an active metamodule or single active module is being used. This allows the

value is still recorded for consistency with the algorithms tested in simulation.

robot representing the active agent to be reused as the next active agent, thus retaining consistency between placements.

Another active agent is initialised at $\{0, 0\}$ when the previous one has been placed, which requires knowledge of when this occurs. It is not considered how this could be obtained, but one possible distributed method could be to pass messages throughout the structure when the previous active agent has placed to inform the module in location $(1, 1)$ to allow another active agent to pass over it.

6.3.2 Bridge Optimisation

The simulated bridge optimisation algorithm allows for multiple active agents concurrently, but the version implemented with the real-world robots only allows one active agent at a time due to the limited number of modules manufactured. The procedure by which placed agents become active is therefore modified so that a single agent is released when the previous active agent has placed. Placed agents with an empty space below and to one side of them are again called release candidates. Whenever there is no active agent and the trial has not finished, a release candidate is selected to be released as follows. For each release candidate j, a value $P_{release,j}$ is calculated according to (4.7) as before, with $\epsilon = 10$, $\mu = 0.3$. However, instead of each release candidate becoming an active agent with this probability, these values are converted into a combined probability mass function $p_{release}(j)$ by dividing each by $\sum_j P_{release,j}$. One release candidate is drawn from this function to become the active agent in the `releasing` mode. This process is implemented using a global controller for simplicity, but message-passing could be used instead to make the algorithm completely distributed. For the case of an active metamodule agent, each module in the metamodule calculates a separate $P_{release,j}$ and if either one is drawn from $p_{release}(j)$ then the corresponding metamodule becomes active. As for the real-world cantilever construction algorithm, the same physical robot is always used as the active agent to retain consistency between placements. Released modules are therefore replaced by this agent.

When an agent is released, it follows the procedure in Algorithm 6.2. This is similar to the version implemented in simulation, but differences arise because of the choices to record H instead of M and F and to only allow one active agent at a time. The active agent records measurements of H to determine whether its release made any links of the module above critical (Line 3) and when recording the distribution of forces around the structure into the $\boldsymbol{\gamma}^{\beta}$ arrays (Line 11).

When the active agent has traversed the whole lower perimeter of the structure and reached column L, it decides whether to remove itself from the environment or place itself to reinforce the structure. If no links are critical, it enters the `escaping` mode and is immediately removed (Lines 22 & 23) instead of slowly passing through the structure as in simulation. If the active agent believes the structure requires reinforcement, $p_{column}(c)$ is calculated from $\boldsymbol{\gamma}^{\beta}$ using the procedure described for the real-world cantilever construction algorithm (Line 16), then it enters the `placing` mode. In this mode, it is not necessary to move the active agent to each column drawn from $p_{column}(c)$ to check whether it is valid, as was the case in simulation with multiple active agents. The active agent instead remembers which columns are valid, and draws columns until a valid c_{target} is chosen. One or two modules are then placed in this column (Lines 25 – 29), depending on whether a

Algorithm 6.2. The bridge optimisation algorithm implemented on the HyMod platform, emphasising differences with the version implemented in simulation (Algorithm 4.1).

```
1  switch mode do
2      case releasing do
3          Record $H$ from placed module above into $\gamma_c^\beta$;
4          if Module above has a critical link then
5              Place back here (agent no longer active);
6              return
7          else
8              mode ← gathering;
9              Make step;
10     case gathering do
11         Record $H$ from placed module above into $\gamma_c^\beta$;
12         if Agent in column $L$ and Previously visited column 1 then
13             if No measured links critical then
14                 mode ← escaping;
15             else
16                 Calculate $p_{column}(c)$ from $\boldsymbol{\gamma}^\beta$;
17                 ~~$c_{target}$ ← sample from $p_{column}(c)$ without replacement;~~
18                 mode ← placing;
19         else
20             Make step;
21     case escaping do
22         Remove agent from structure;
23         return
24     case placing do
25         while Not placed do
26             $c_{target}$ ← sample from $p_{column}(c)$ without replacement;
27             if Valid location then
28                 Place in column $c_{target}$;
29                 return
30     ~~case swapping do ...~~
```

single active module or active metamodule is being used. The `swapping` mode and the timeout are not necessary.

Once an active agent is placed or removed from the structure, another agent is released. This continues until either there are no more release candidates, indicating the structure consists of a single row of agents, or an active agent chosen to be released is in the same location the previous one placed in, indicating the shape of the structure has settled. This is different to the simulated bridge optimisation algorithm, in which the trial finished when 20 timesteps passed without any change to the structure. The metric is changed to reduce the time taken to perform each trial.

6.3.3 Bridge Deconstruction

In the simulated version of the bridge deconstruction algorithm, modules are added to the cantilever from above the left support, and are removed during bridge deconstruction above the right support; it is assumed that the goal of the bridge is to allow modules to travel from the left side of the gap to the right side. However, due to constraints with the fixed supports manufactured (Section 6.1.4), in the real-world implementation the algorithm both adds and removes modules above the left support. It is therefore instead assumed that the task on the right side of the gap is completed by a subset of the available modules, who then travel back over the bridge.

During real-world bridge deconstruction, only one active agent can move around the structure at a time, as for the other real-world algorithms. Active agents arise by one of three methods:

1. A new active agent can initialise in position $\{0, 1\}$.
2. An existing placed agent in the structure can release itself.
3. The current active agent can flag a placed agent to be released, emulating the `force-releasing` mode for this scenario with only a single active agent at a time.

If an agent is flagged for release, it is released at the next opportunity. In order to choose between the other two options, $P_{release,j}$ is first calculated for each release candidate j in the same manner as the simulated algorithm. Only modules right of the narrow section are considered release candidates as those on the left are likely in a useful position already, the mirror of the simulations. Each release candidate j is selected to release with its $P_{release,j}$: this is similar to the simulated bridge deconstruction algorithm, but different to the real-world bridge optimisation algorithm. If multiple release candidates are chosen at once, then one is selected at random to release. If no release candidates are chosen then a new active agent initialises in $\{0, 1\}$ instead. As in the previous real-world algorithms, adding new active agents is achieved through global control, but could be made distributed in future. In addition, the same physical robots are used as the active agent in each trial for consistency, which replace the modules that are released from the structure.

The procedure followed by the active agent during the bridge deconstruction algorithm is shown in Algorithm 6.3. Choosing to measure H instead of M and F (Lines 5 & 13) results in $p_{column}(c)$ being calculated from the modified data $\boldsymbol{\gamma}^{\beta}$ in Line 15, similar to the other real-world algorithms. This choice also means the raw values of H are used to determine the stability of the structure in Line 16 instead of attempting to approximate

Algorithm 6.3. The bridge deconstruction algorithm implemented on the HyMod platform, emphasising differences with the version implemented in simulation (Algorithm 5.1).

```
switch mode do
    ~~case force-releasing do~~
        ~~...~~
    case releasing do
        Record $H$ from placed module above into $\gamma_c^\beta$;
        if Module above has a critical link then
            Place back here (agent no longer active);
            return
        else
            mode ← gathering;
            Make step;
    case gathering do
        Record $H$ from placed module above or below into $\gamma_c^\beta$;
        if Agent in column 1 then
            Calculate $p_{column}(c)$ from $\boldsymbol{\gamma}^\beta$;
            if No measured links critical then
                Flag placed agent at bottom of column $c_{release}$ for release;
            mode ← placing;
        else
            Make step;
    case escaping do
        if In column 1 then
            Flag next placed agent for release;
            Agent leaves environment;
            return
        Make step;
    case placing do
        while Not placed do
            $c_{target}$ ← sample from $p_{column}(c)$ without replacement;
            if Valid location then
                Place in column $c_{target}$;
                return
    ~~case swapping do ...~~
```

the values in the equivalent unsupported cantilever as described in Section 5.2.2. If the active agent believes the structure is stable upon gathering all the necessary data, an agent is flagged for release without moving the active agent there as was required in simulation (Line 17). This agent is the one at the bottom of column $c_{release}$, where:

$$c_{release} = \begin{cases} c_{n,r} & \text{if } c_{n,r} = L \\ (c_{n,r} + 1) & \text{otherwise} \end{cases} \tag{6.4}$$

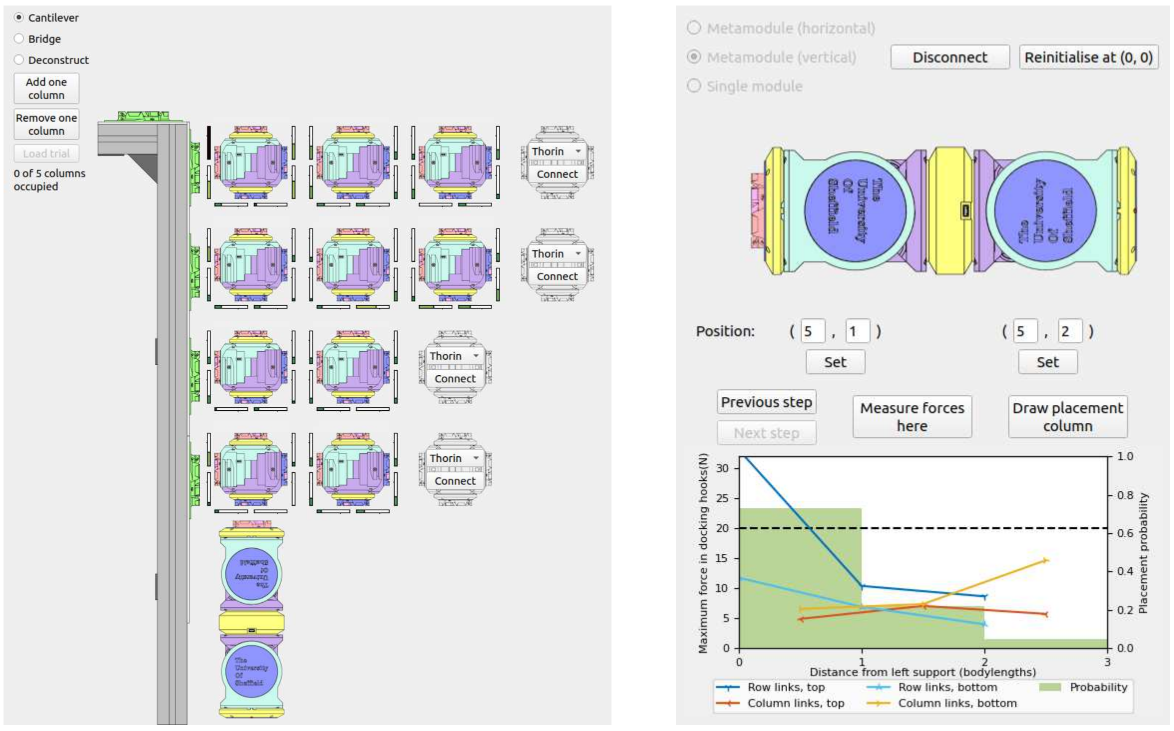
where $c_{n,r}$ denotes the rightmost column in the narrow section. Flagging agents for release in this global manner removes the need for the `force-releasing` mode used in simulation (Lines 2 – 3). Regardless of whether an agent is flagged for release at this time, the active agent enters the `placing` mode. As in the previous real-world algorithms, it then repeatedly samples from $p_{column}(c)$ without replacement until a valid location is chosen: it is only moved when a valid column is drawn (Lines 28 – 32).

This process repeats until the agent attached to the right support is released. The removal phase then begins, in which agents are removed columnwise from the top of each column. In the simulated algorithm, placed agents are switched to the `force-releasing` mode as active agents pass over them in this phase. This is simplified in the real-world implementation: the agent at the top of the rightmost column is released and enters the `escaping` mode when the previous active agent reaches the top of column 1 and exits the environment (Lines 22 – 25). As was the case when active agents are added during the reinforcement phase, this requires global control. However, it could be made distributed by using the module in position $(1, 1)$ to send messages to other placed modules instructing them to release when the previous active agent has passed over it. Due to the simplified scenario, neither the `swapping` mode, the timeout period, nor the additional behaviours based on the position the active agent ends the timestep in, are required.

6.4 Real-World Experimental Results

6.4.1 Experimental Procedures

To perform the trials of the algorithms, a custom Qt GUI consisting of two windows was created with Python, shown in Figure 6.17. One window shows a digital twin of the current structure (Figure 6.17a). Modules communicate with a central computer through BLE which displays this GUI to the user. Inter-agent communication is simulated by this central computer, instead of requiring individual modules to communicate with each other through Bluetooth. Implementing the algorithms in this way makes trials faster to run as modules do not need to continuously pair and unpair with each other when communicating. It also allows more data to be presented to the user during the trial and saved for later analysis. The user can arrange the passive HyMod modules in a configuration reflecting the current state of the structure. They can also see the calibrated force measurements in each strain gauge within the structure, and actuate the connectors of each module. A second window shows the active agent (Figure 6.17b). The user can control its connectors and move it around the structure. As it is moved, measurements of H recorded by the passive modules in the structure are recorded and plotted in the graph at the bottom of the window, which also shows the current probability mass function $p_{column}(c)$.



(a) **(b)**

Figure 6.17. The digital twin GUI used during real-world experiments, shown for cantilever construction by an active metamodule. **(a)** The main window visualising the current state of the structure, allowing the user to control and stream data from the placed agents. **(b)** The active agent window, through which the user controls the active agent to move around the structure. The graph at the bottom of **(b)** shows the measurements of H recorded in real-time as the active agent is moved around the structure, and the resulting probability mass function $p_{column}(c)$.

Each algorithm was tested for $H_{allowable} \in \{16, 23\}$ N, chosen as three and four standard deviations respectively from the mean strength of a single docking hook found in Section 6.2.1. This allows differences in behaviour between tests with strong and weak links to be observed. Separate trials were performed with an active metamodule to demonstrate how a fully-autonomous system would function, and with a single active module to build higher-resolution structures with the limited number of modules manufactured. Ten trials of each algorithm were performed for each $H_{allowable}$ and both types of active agent. Each time the algorithm chose to place a module, it was randomly selected from the remaining unplaced modules[2]. Whenever the active agent recorded measurements of H from an adjacent module, the structure was allowed to settle for at least 2 s after instructing measurements to be recorded before they were actually taken. The locations of all modules in the structure and the calibrated force measurements in each strain gauge within their links were also saved at these times for later analysis.

[2]Two modules had small electrical faults meaning one strain gauge did not work reliably. While the module selection was random, these were always the last two to be placed.

Algorithm	**Active agent mode**	$H_{allowable}$ **(N)**	L **(bodylengths)** 2	3	4	5
Bridge optimisation	Metamodule	16	✗	✓	✗	✗
		23	✗	✓	✓	✗
	Single module	16	✓	✓	✗	✗
		23	✗	✓	✓	✓
Bridge deconstruction	Metamodule	16	✗	✓	✗	✗
		23	✗	✓	✓	✗
	Single module	16	✗	✓	✗	✗
		23	✗	✓	✓	✓

Table 6.1. The different combinations of active agent mode, $H_{allowable}$, and L the bridge optimisation and deconstruction algorithms were tested for.

6.4.1.A Cantilever Construction Procedure

Trials of the cantilever construction initialised with one or two randomly selected modules in location $\{1, 1\}$, depending on the type of active agent. Trials finished when all the modules were placed, with the active agent itself added to the structure in the final step. Trials with an active metamodule therefore consisted of five placement decisions and finished with a structure of twelve modules, while trials with a single active module consisted of nine placement decisions and finished with a structure of ten modules. When the active metamodule was moved around the structure, it was oriented such that rotating about each body joint would allow it to move to the next position through its flipping gait. In contrast, the single active module was attached in the same orientation as the placed agents.

The trials were performed in two groups, with the batteries of all modules fully charged before each group. The first group consisted of the trials with an active metamodule, and the second group consisted of the trials with a single active module. Within each group, trials were alternated between $H_{allowable} = 16\,\text{N}$ and $H_{allowable} = 23\,\text{N}$ to mitigate against any sensor drift or changes in environmental conditions that might occur during the trials.

6.4.1.B Bridge Optimisation Procedure

Each trial of the bridge optimisation algorithm began with a structure randomly chosen from a trial of the cantilever construction algorithm when it first reached a specified L. The algorithm was tested for the combinations of active agent mode, $H_{allowable}$, and L shown in Table 6.1. These are all the lengths reached during these trials before the active agent was placed for $L \geq 3$ bodylengths, as well as for a single active module with $H_{allowable} = 16\,\text{N}$ and $L = 2$ bodylengths. Usually, the bridge optimisation algorithm is not tested for $L < 3$ bodylengths as the cantilever construction algorithm produced stable bridges with at most one agent below row 1, so the optimisation would simply remove this agent: structures with more agents are produced by a single active module for $H_{allowable} = 16\,\text{N}$ and $L = 2$ bodylengths, so the algorithm was also tested for these parameters.

Once a trial was selected, the necessary modules were placed in their corresponding positions, and the rightmost ones were also connected to the freestanding right support.

Agents selected for removal were replaced by the active agent before any forces were read. The orientation of the active metamodule was as before, but the single active module was oriented perpendicular to the placed agents. This allowed it to fit more easily in gaps a single column wide, which was otherwise tricky in locations with placed agents on both sides due to the permanently extended passive HiGen connectors.

As for the cantilever construction algorithm, trials of the bridge optimisation algorithm were performed in two groups split by the type of active agent, and the batteries were fully charged before each group. Within each group, trials again alternated in a consistent manner. One trial for each length with $H_{allowable} = 16\,\mathrm{N}$ was performed in increasing order of L, then one trial for each length with $H_{allowable} = 23\,\mathrm{N}$ was performed in increasing order of L. This order repeated until ten trials of each combination were completed.

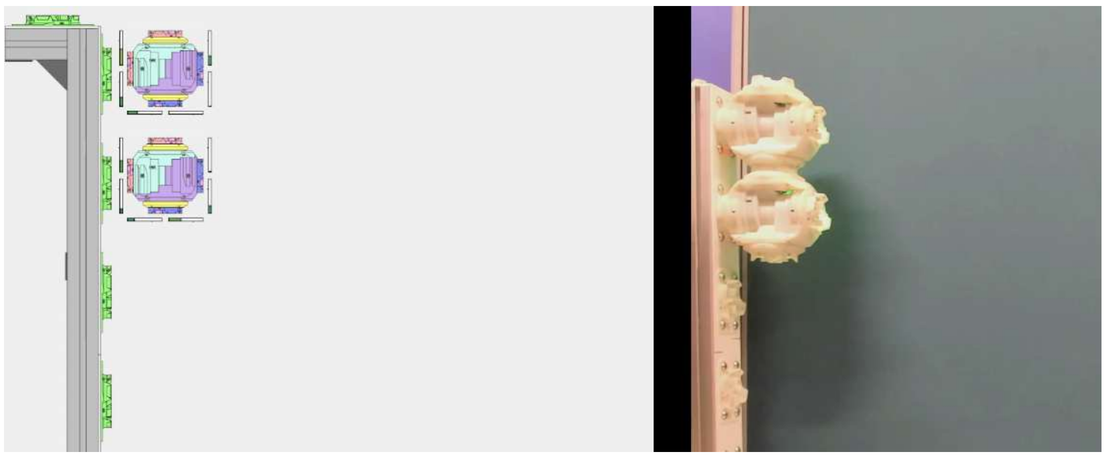
6.4.1.C Bridge Deconstruction Procedure

Each trial of the bridge deconstruction algorithm began with a structure randomly chosen from a trial of the bridge optimisation algorithm of the specified type of active agent, $H_{allowable}$, and L. The bridge deconstruction algorithm was tested for all combinations of $H_{allowable}$ and L that the bridge optimisation algorithm was tested for, with the exception of bridges of $L = 2$ bodylengths (Table 6.1) which were trivial to deconstruct: such a short bridge was always found to be stable, so the active agent would always immediately place in column 1 and initiate the removal phase. Once a trial was randomly selected, the required modules were replaced in their locations to create the necessary bridge. Trials were performed in a similar order to the bridge optimisation algorithm.

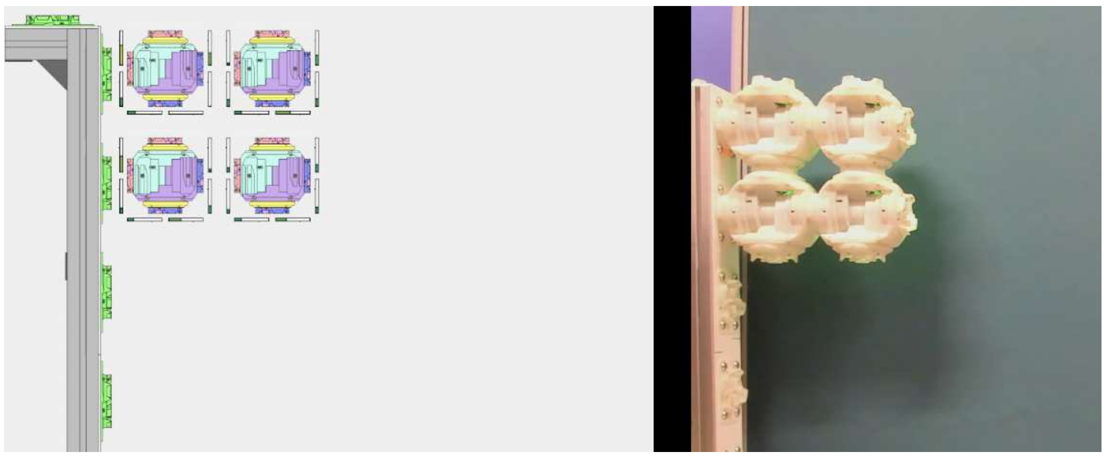
6.4.2 Cantilever Construction Results

The cantilever construction algorithm successfully produced a stable structure in all trials, with no link failures occurring. This demonstrates the advantages of using the distributed force-aware construction approach of this algorithm. Frames from example construction sequences are shown in Figures 6.18 & 6.19 for cantilevers built by an active metamodule and single active module respectively. These figures show how agents are added to the structure, but do not include the movements of the active agent around the structure. Agents are seen adding at the tip of the structure if the measurements of H indicate that it is stable, and otherwise reinforcing the structure at the bottom of an existing column. Sometimes the weight of the active agent causes the structure to become unstable while in the `gathering` mode, leading to a structure that appears stable being reinforced instead of extended: this occurs when the structures shown in Figures 6.19a & 6.19e are added to. These trials show the longest cantilevers built across all the trials for each type of active agent. Note how the higher-resolution structure built by the single active module can extend further than those built by the active metamodule, as modules can be placed in more efficient positions.

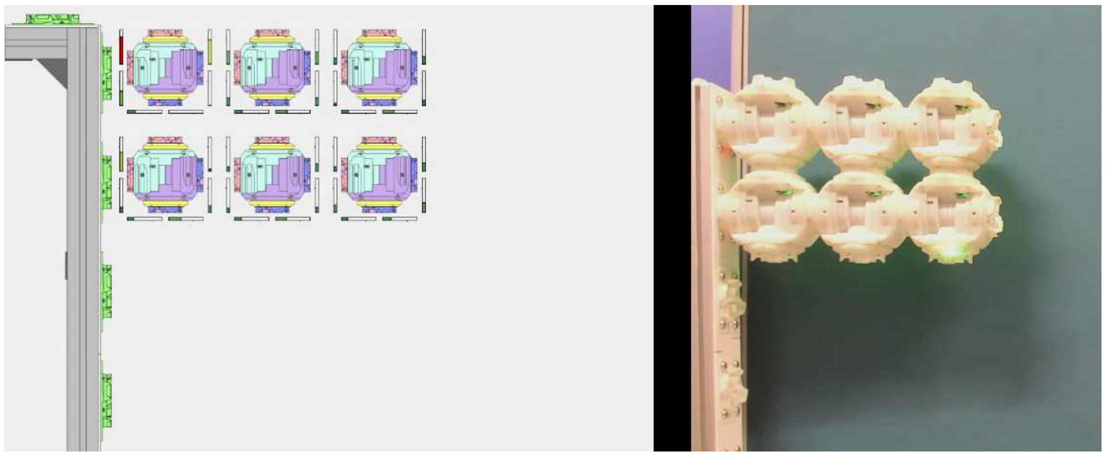
The evolution of the structure as the cantilever construction algorithm runs for each trial is presented in Figures 6.20 & 6.21. These figures show which structures were built, how often each was built, and where the active agents chose to add themselves to each structure. They show that using a single active module results in a higher degree of variability than an active metamodule, which is to be expected as more choices can be



(a)



(b)



(c)

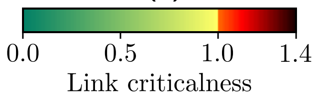
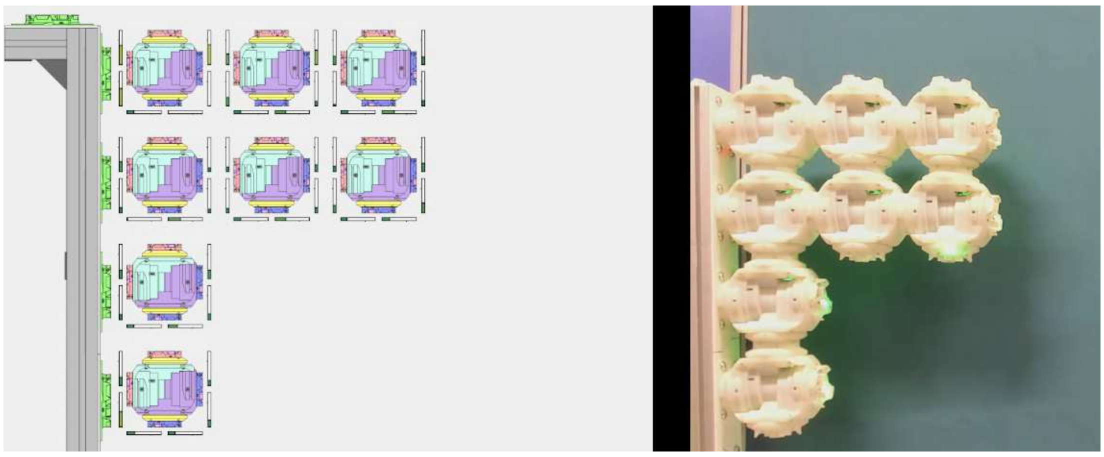
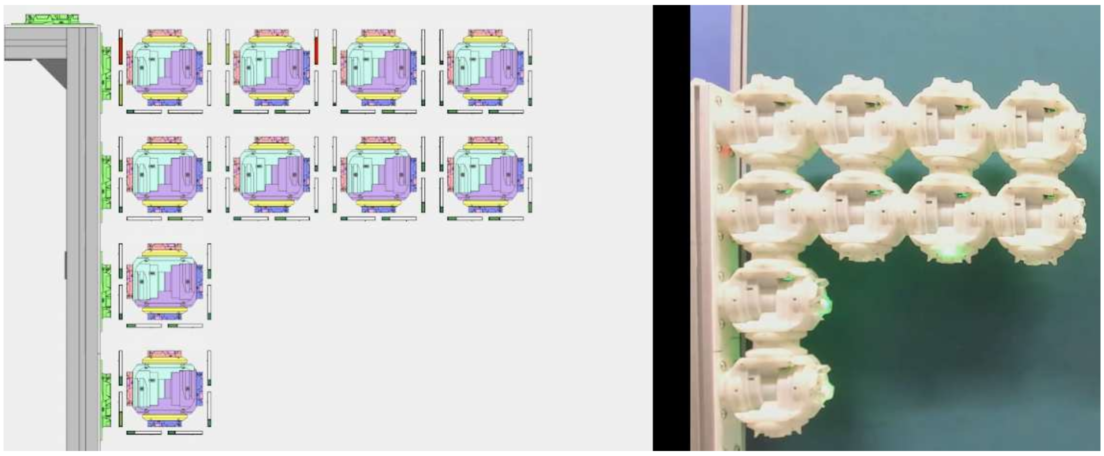


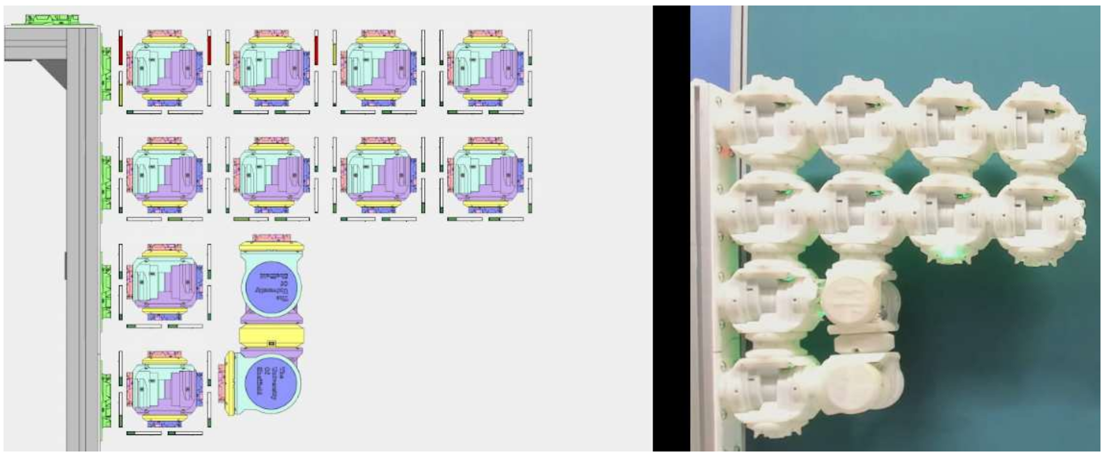
Figure 6.18. Frames from an example cantilever construction sequence with an active metamodule for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



(d)



(e)



(f)

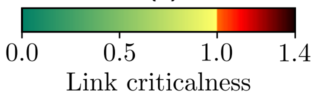
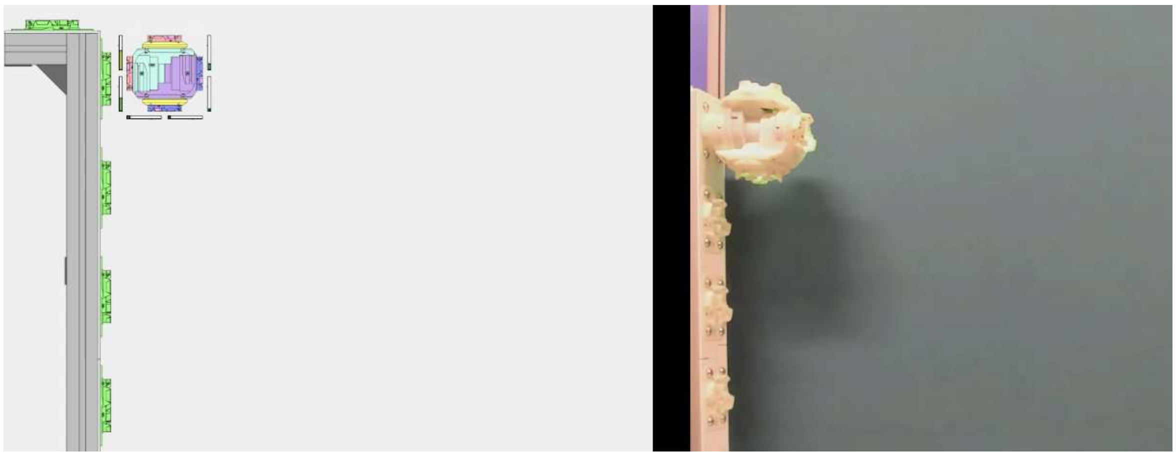
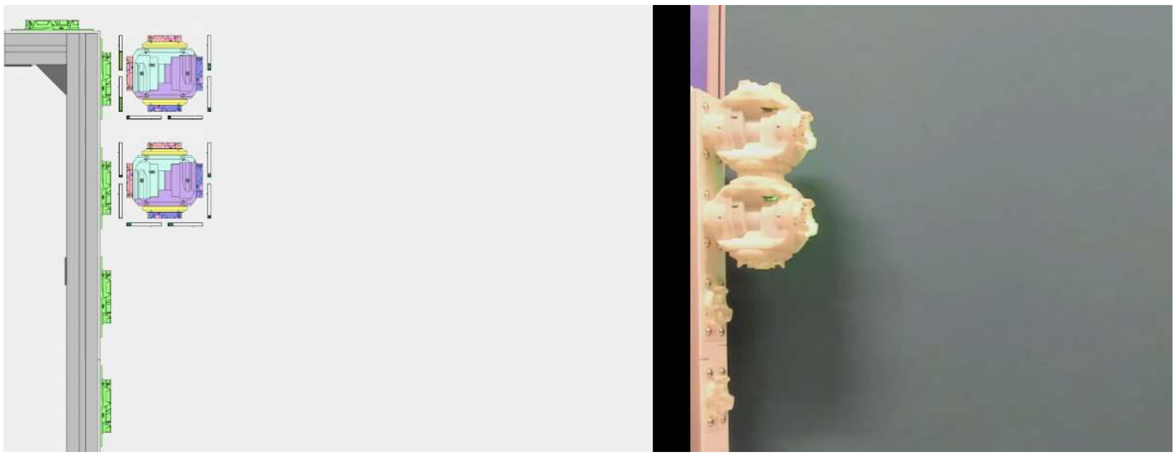


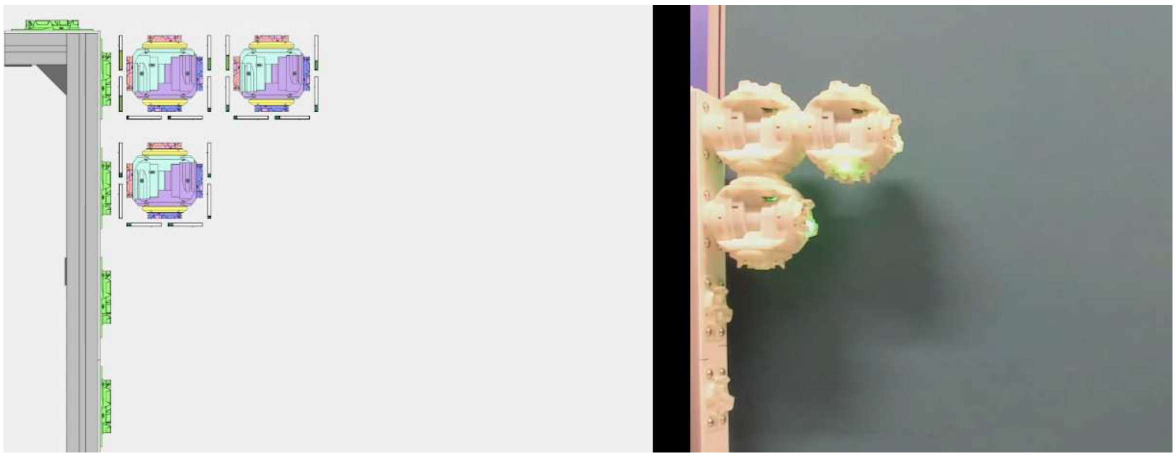
Figure 6.18. [continued] Frames from an example cantilever construction sequence with an active metamodule for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



(a)



(b)



(c)

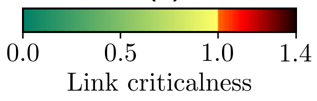
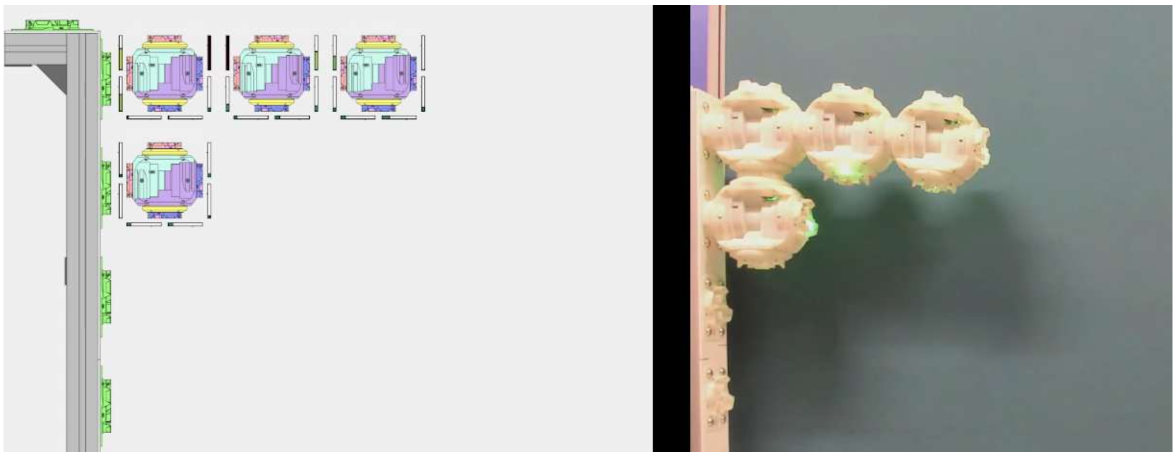
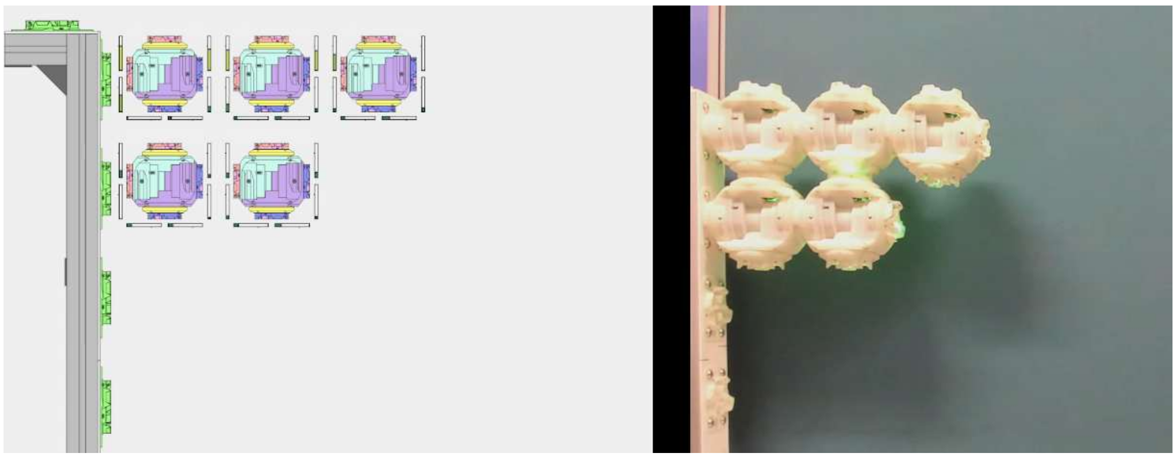


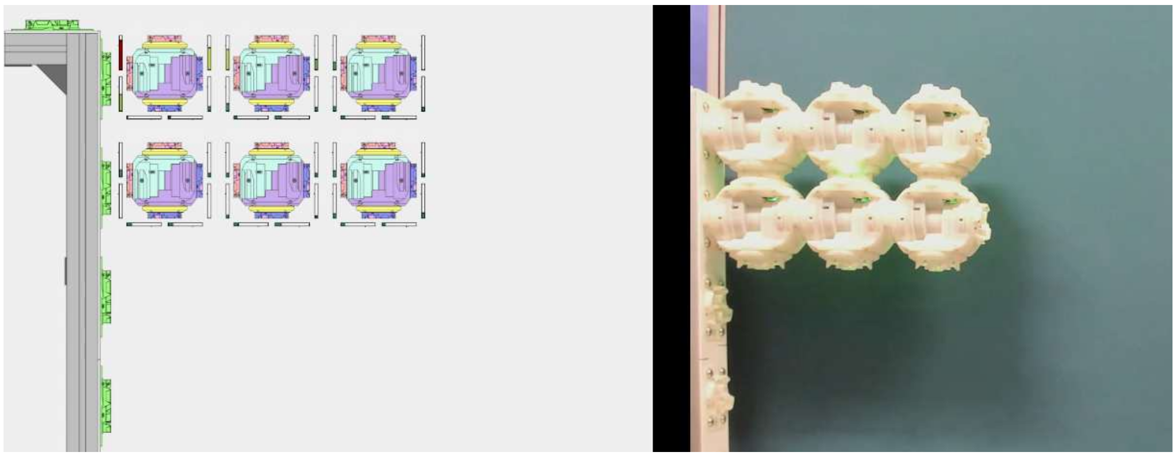
Figure 6.19. Frames from an example cantilever construction sequence with a single active module for $H_{allowable} = 23\,\text{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



(d)



(e)



(f)

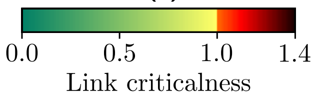
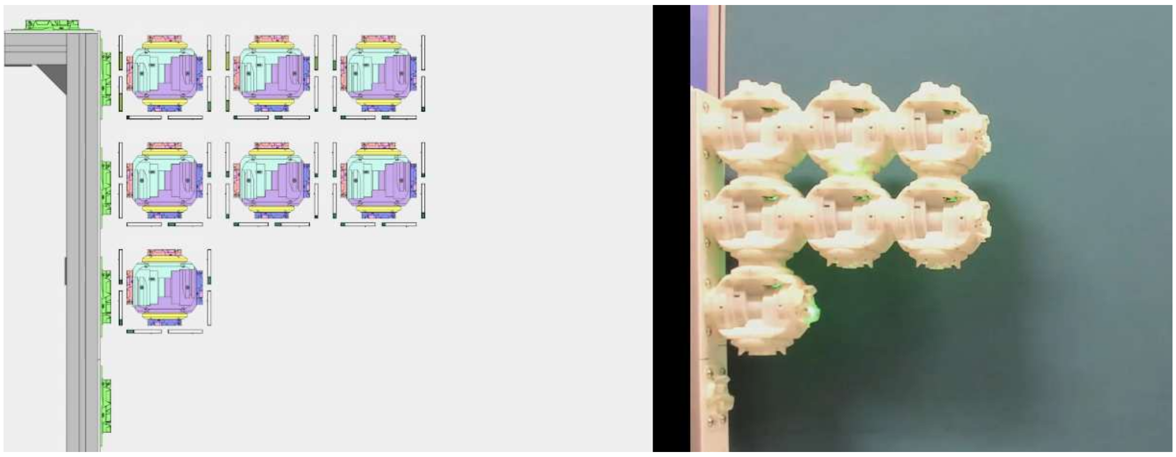
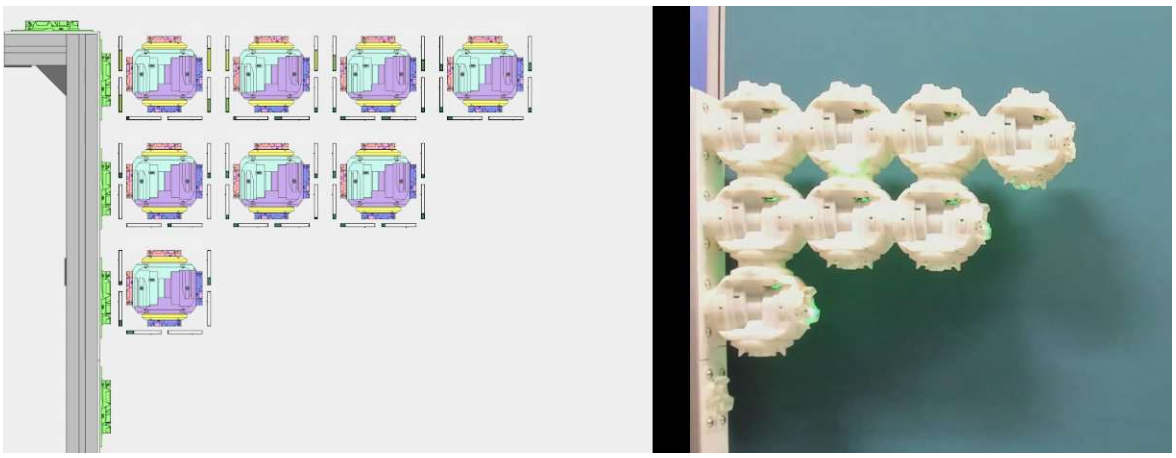


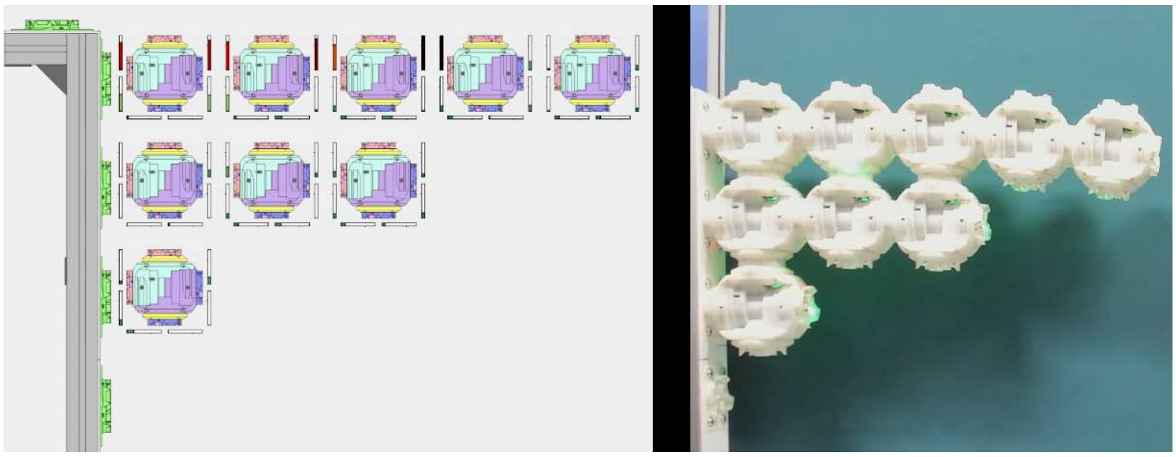
Figure 6.19. [continued] Frames from an example cantilever construction sequence with a single active module for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



(g)



(h)



(i)

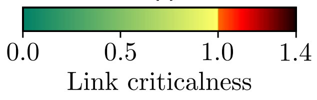


Figure 6.19. [continued] Frames from an example cantilever construction sequence with a single active module for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.

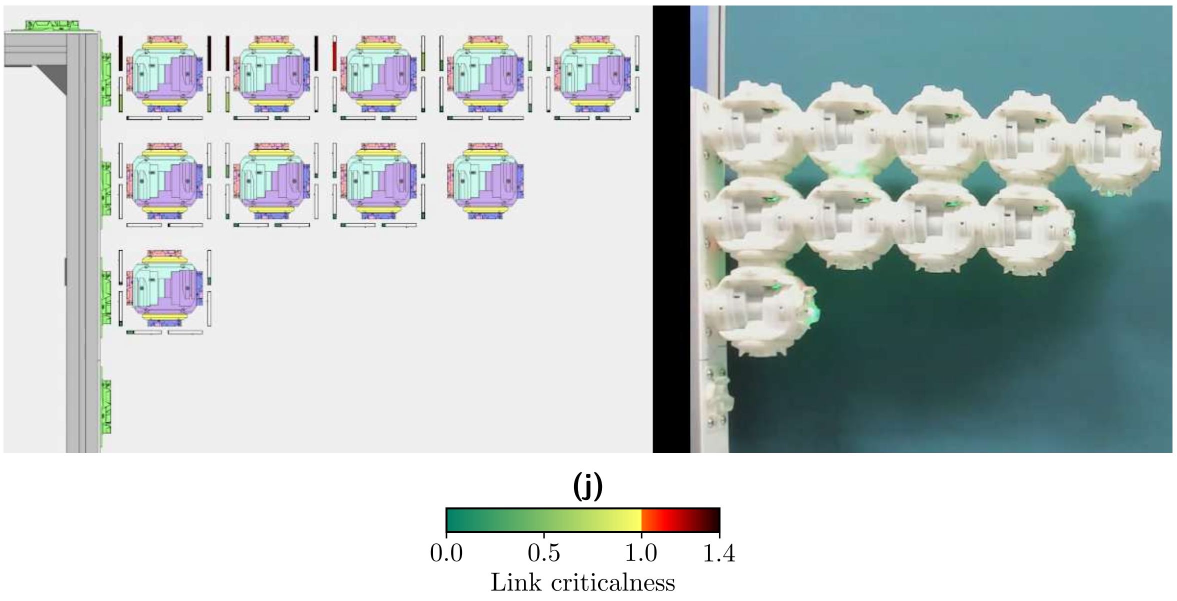


Figure 6.19. [continued] Frames from an example cantilever construction sequence with a single active module for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.

made with the limited number of modules. When a higher $H_{allowable}$ is used, the algorithm builds longer and more slender structures for both kinds of active agent.

Graphs summarising the results of the trials are shown in Figures 6.22 & 6.23 for an active metamodule and single active module respectively. As expected, Figures 6.22a & 6.23a show that longer cantilevers can be built if $H_{allowable}$ is greater. The active metamodule could build structures up to 4 bodylengths long, whereas the single active module reached five bodylengths in 30 % of trials with $H_{allowable} = 23\,\mathrm{N}$. This indicates that the higher granularity of structure built by adding only one module at a time allows for the modules to be used in a more efficient manner to provide support where required.

Histograms showing the maximum γ measured across all the strain gauges in the structure at each step are shown in Figures 6.22b & 6.23b. It can be seen that for the active metamodule, $H_{allowable}$ was exceeded up by to two times, whereas for a single active module it was occasionally exceeded by twice this. The active metamodule therefore produces more conservative structures than a single active module. Using a higher $H_{allowable}$ led to structures with a lower mean maximum γ in the active metamodule trials, but to structures with a higher mean maximum γ in the single active module trials; however, the difference in the latter case is not statistically significant according to a Mann-Whitney U test ($p = 0.79$). Furthermore, trials for both active agents with higher $H_{allowable}$ also had a higher proportion of steps in which the structure was stable. This indicates that a higher $H_{allowable}$ leads to a '*high risk, high reward*' construction sequence: structures get close to failure, but only for short periods of time, after which they are successfully reinforced.

In order to build the longer structures observed for higher $H_{allowable}$, the algorithm is more likely to produce more slender structures (Figures 6.22c & 6.23c). The size of the vertical fixed support limits the number of rows that are allowed in the structure,

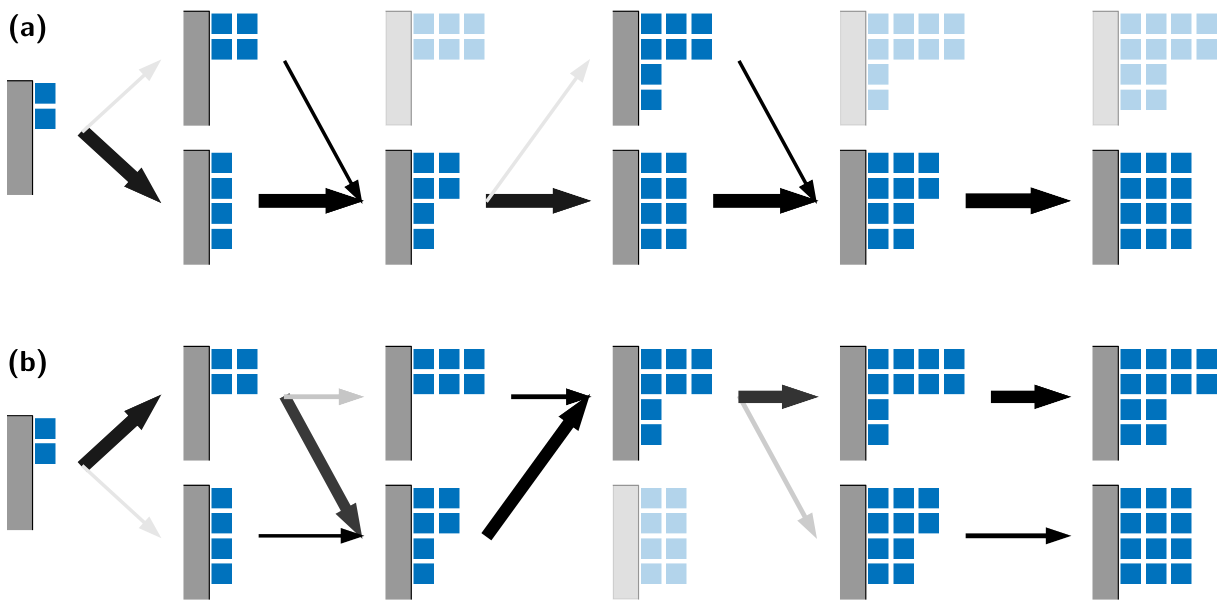
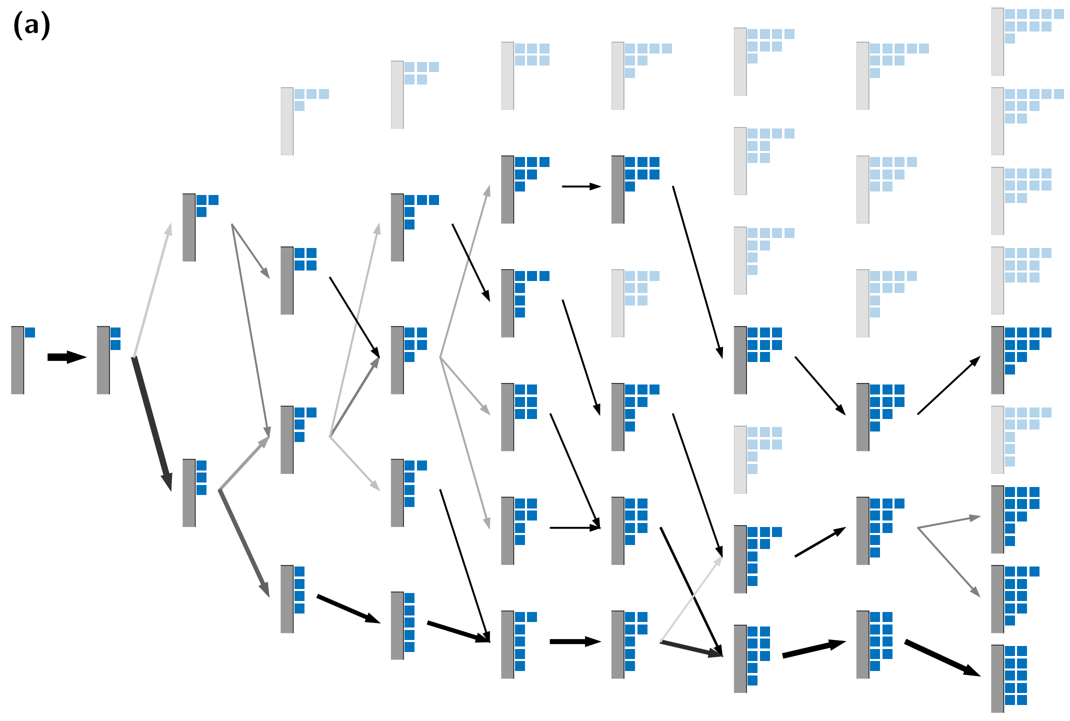


Figure 6.20. Progression of the cantilever construction algorithm for structures built by the active metamodule for **(a)** $H_{allowable} = 16\,\mathrm{N}$, and **(b)** $H_{allowable} = 23\,\mathrm{N}$. Modules are shown in blue and the fixed support in grey. Each column contains structures with an equal number of agents, with longer structures at the top of each column. Structures that are only built by the other $H_{allowable}$ are shown paler. The width of arrows is proportional to the number of trials that take this choice, and the colour is darker proportional to the percentage of trials in which that configuration was built which took that particular path.



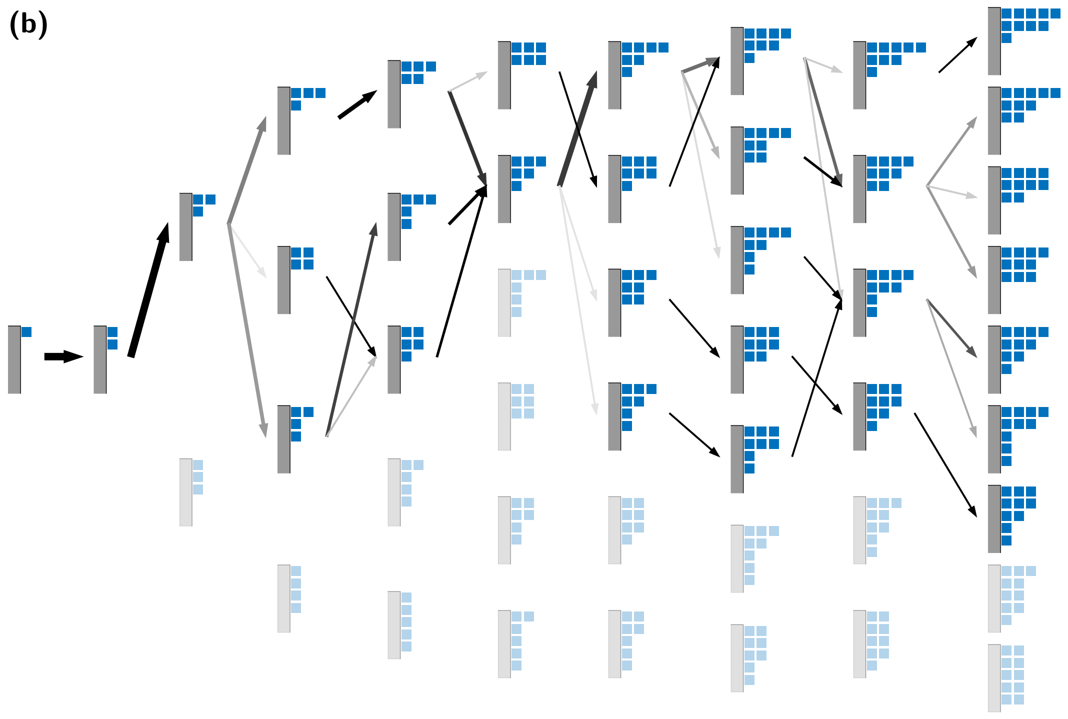
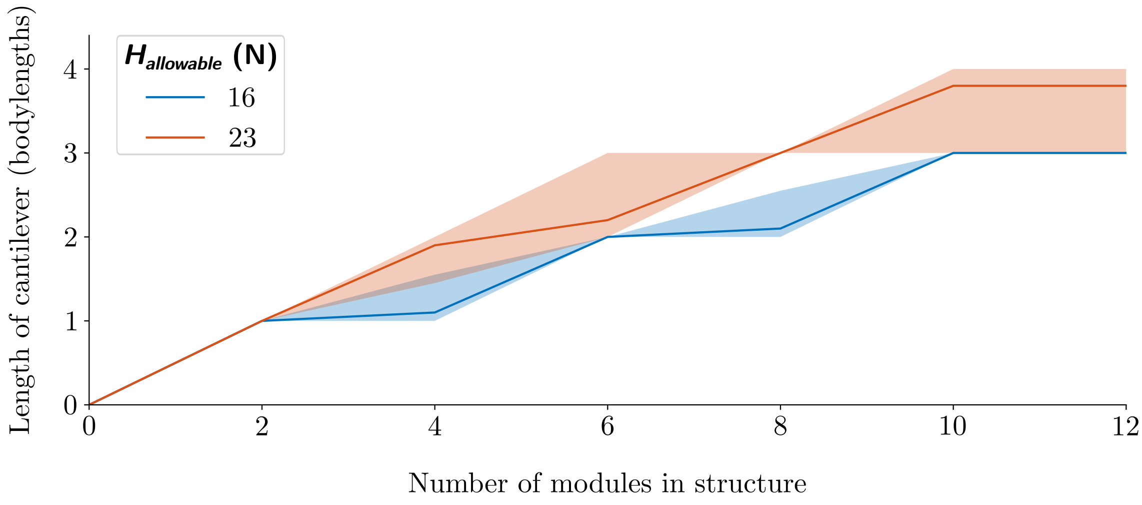
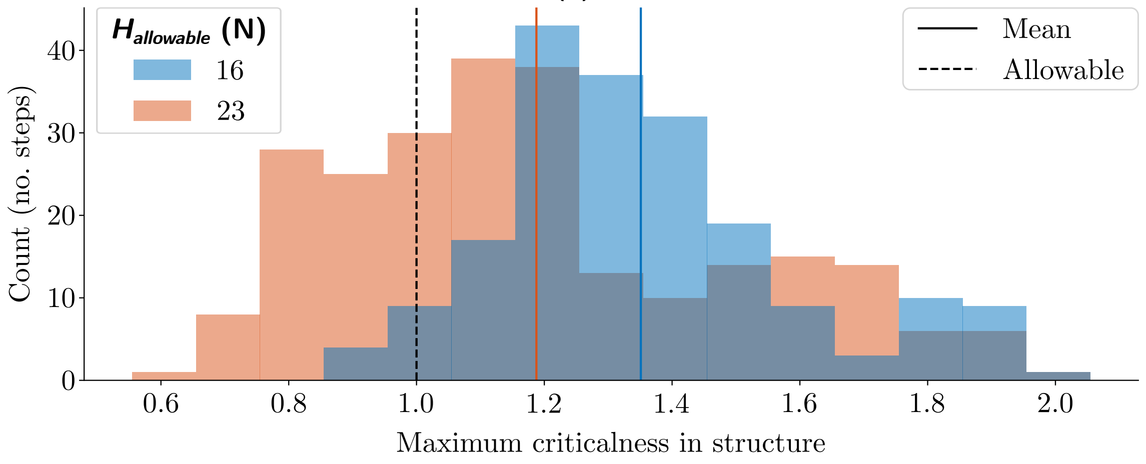


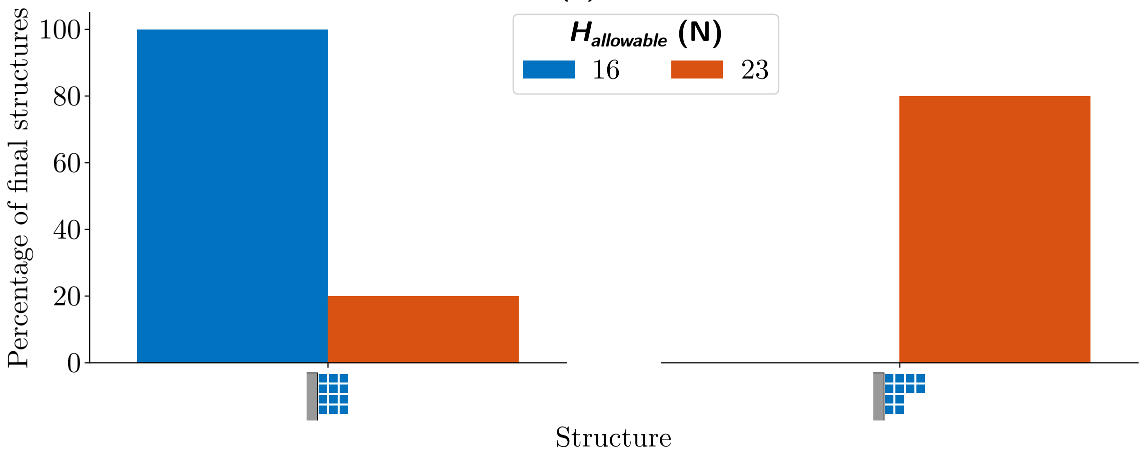
Figure 6.21. Progression of the cantilever construction algorithm for structures built by the single active module for **(a)** $H_{allowable} = 16\,\text{N}$, and **(b)** $H_{allowable} = 23\,\text{N}$. Modules are shown in blue and the fixed support in grey. Each column contains structures with an equal number of agents, with longer structures at the top of each column. Structures that are only built by the other $H_{allowable}$ are shown paler. The width of arrows is proportional to the number of trials that take this choice, and the colour is darker proportional to the percentage of trials in which that configuration was built which took that particular path.



(a)

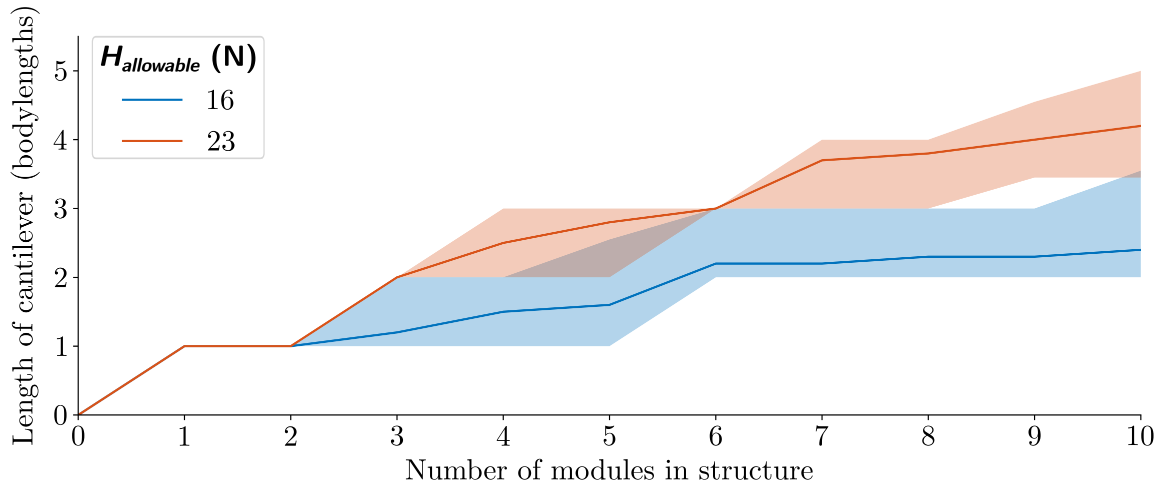


(b)

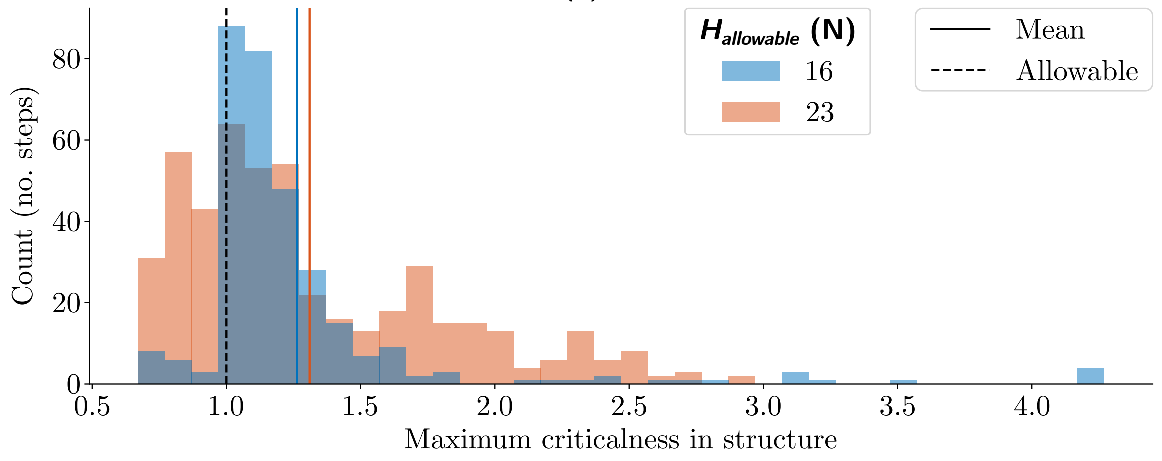


(c)

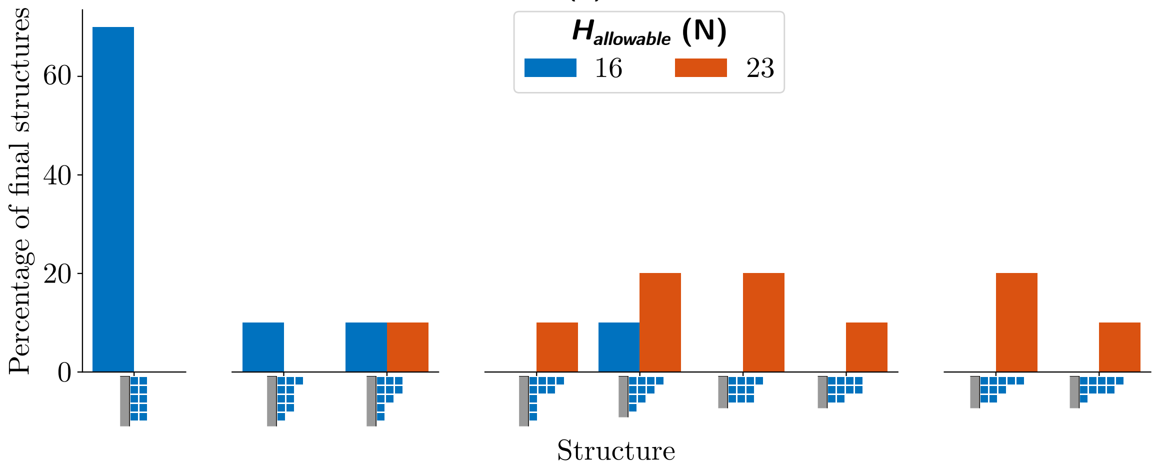
Figure 6.22. Results from the cantilever construction algorithm with the active meta-module. **(a)** The length of the structure as more modules are added. **(b)** A histogram showing the number of steps in which the maximum criticalness measured by any strain gauge in the structure was as given. **(c)** The percentage of trials in which different final structures are built. Lines in **(a)** show mean values with the area between the 5 % and 95 % quantiles shaded, and the histogram bin width in **(b)** is 0.1.



(a)



(b)



(c)

Figure 6.23. Results from the cantilever construction algorithm with the single active module. **(a)** The length of the structure as more modules are added. **(b)** A histogram showing the number of steps in which the maximum criticalness measured by any strain gauge in the structure was as given. **(c)** The percentage of trials in which different final structures are built. Lines in **(a)** show mean values with the area between the 5 % and 95 % quantiles shaded, and the histogram bin width in **(b)** is 0.1.

meaning the active metamodule could only produce two structures, whereas the single active module produced nine. The structures get thinner towards the tip, as specified by the continuity condition built-in to the self-assembly algorithm.

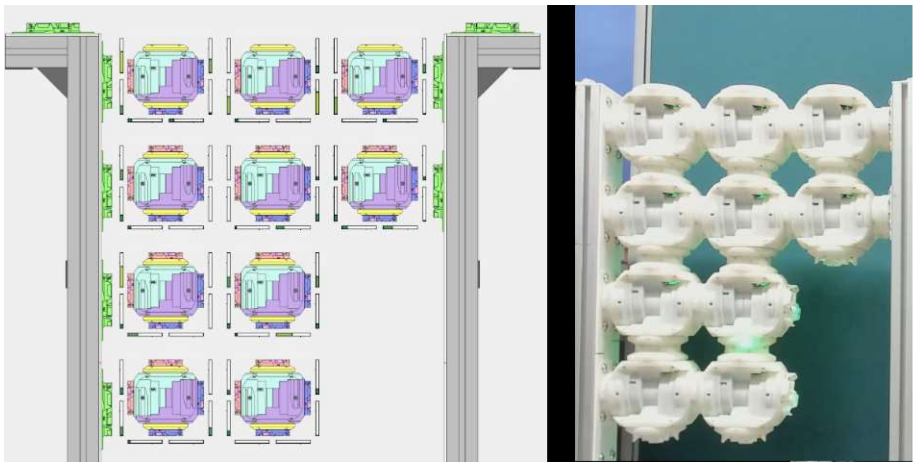
6.4.3 Bridge Optimisation Results

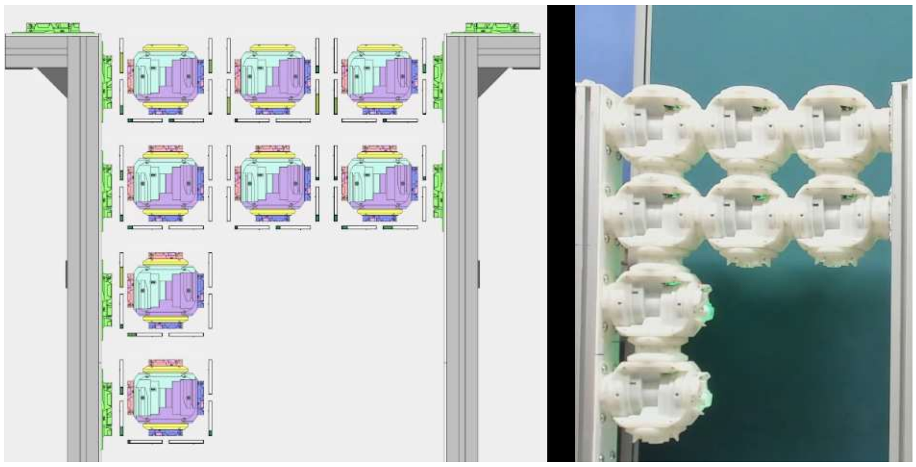
All trials of the bridge deconstruction algorithm resulted in the structure reconfiguring without any links failing. This process is illustrated for an active metamodule in Figure 6.24, showing how the number of modules is reduced. In this example, the structure is stable at all stages of the optimisation, and the final bridge contains the minimum number of modules required to span the gap when an active metamodule is used.

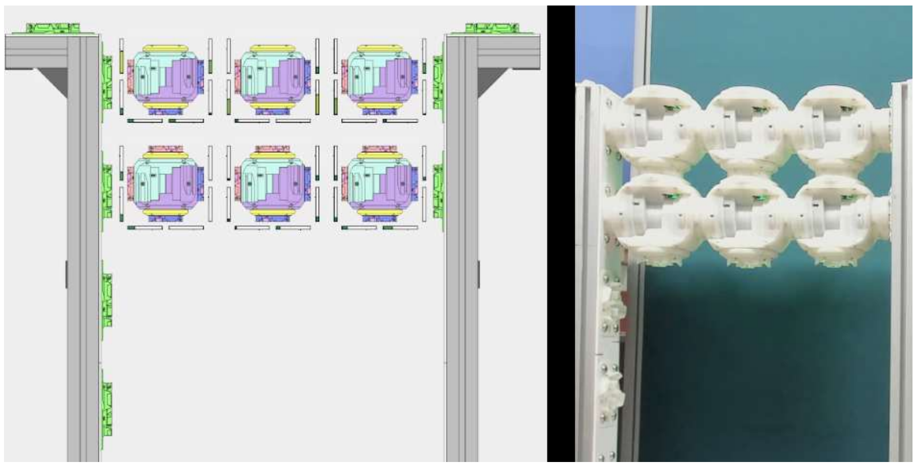
In almost all trials, the number of modules in the structure was reduced as the algorithm progressed. There was one exception to this, which occurred during a trial with a single active module, $H_{allowable} = 23\,\mathrm{N}$, and $L = 4$ bodylengths. The reconfiguration sequence for this trial is shown in Figure 6.25. The link between positions $(1, 2)$ and $(1, 3)$ is always reported to be critical. This causes modules to be repeatedly moved from the left to the right of the narrow section as they attempt to provide support to this link. The trial eventually finishes when the agent in position $(2, 3)$ repeatedly releases and places back where it was, as doing so makes the link on the left of the module above critical. The most likely reason for this anomalous behaviour is that the right support was not positioned correctly before beginning the trial, introducing a horizontal tension into the structure that cannot be reduced by repositioning modules. However, this trial illustrates how the algorithm allows agents to be moved to the right side of the narrow section, and highlights possible issues that a fully-autonomous system might face.

The results from all trials of the bridge optimisation algorithm are summarised in Figures 6.26 & 6.27. There are similar trends for trials with both types of active agent, where the number of modules decreases as each trial continues. For structures of the same length, a higher $H_{allowable}$ leads to a faster rate of module removal and a corresponding lower final number of modules (Figures 6.26a & 6.27a). Longer bridges with the same $H_{allowable}$ required more modules in the final structure to remain stable. The trials with an active metamodule and $H_{allowable} = 23\,\mathrm{N}$ all progressed in the same way for each structure length, with the same number of agents at each step. There were two trials with a single active module for $L = 3$ bodylengths, $H_{allowable} = 16\,\mathrm{N}$ in which the structure was initialised with eight modules, but none were successfully released, resulting in the high 95 % quantile on the corresponding plot in Figure 6.27a. The starting bridge in these trials had 3 modules in row 1, 2 modules in row 2, and 1 module in rows $3 - 5$. The module in position $(4, 1)$ had a critical link on its left side, so when the module below released itself, it was replaced immediately due to this critical link. One trial with the same starting configuration was able to optimise until only three modules remained, demonstrating how the same initial configuration can lead to different outcomes due to real-world inconsistencies.

Figures 6.26b & 6.27b show the maximum γ measured by strain gauges across the whole structure at each step during bridge optimisation. The average maximum γ was found to be considerably lower than that recorded during the initial cantilever construction. Furthermore, the bridges were rarely unstable, whereas critical links were common during cantilever construction. These results indicate that the right support provides sig-


(a)


(b)


(c)

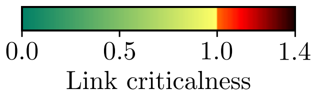
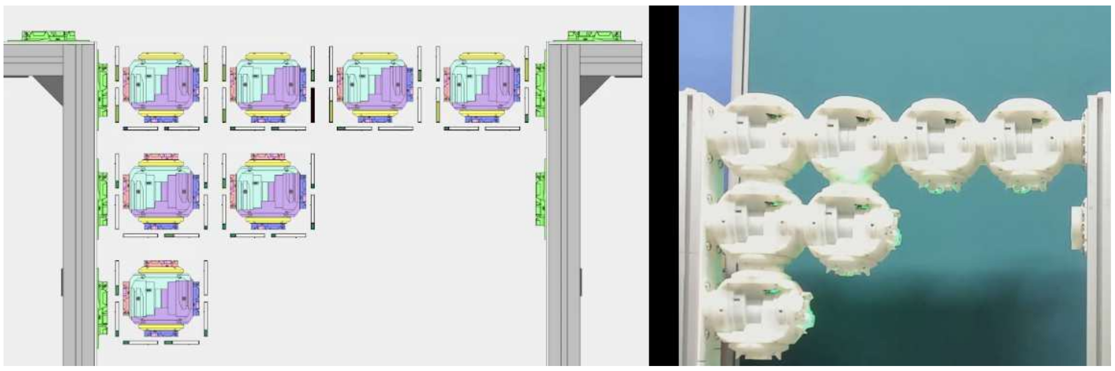
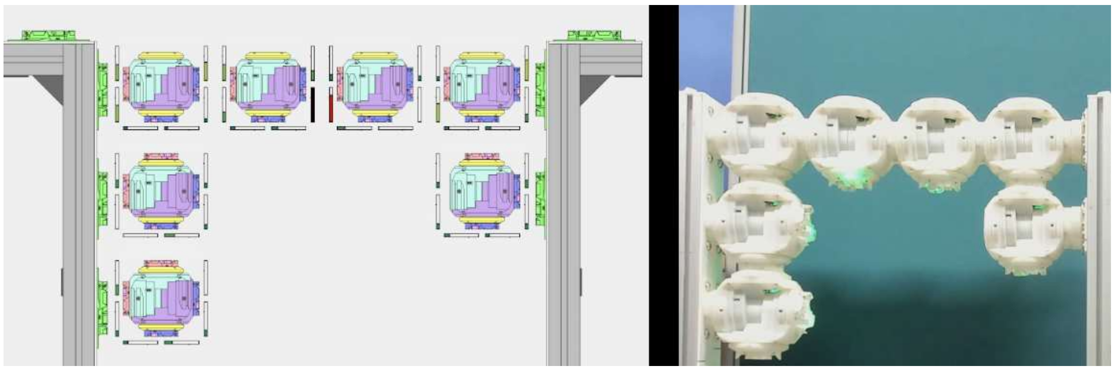


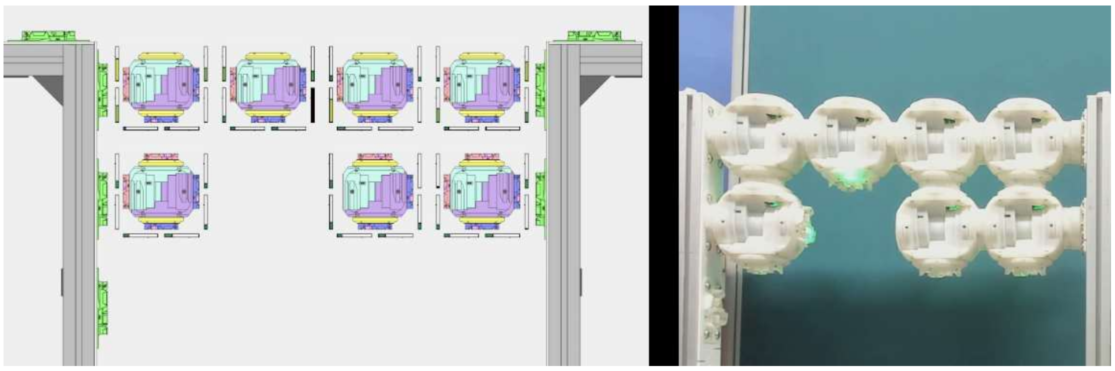
Figure 6.24. Frames from an example bridge optimisation sequence with an active metamodule for $H_{allowable} = 16\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



(a)



(b)



(c)

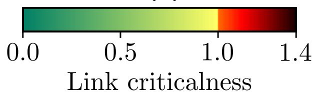
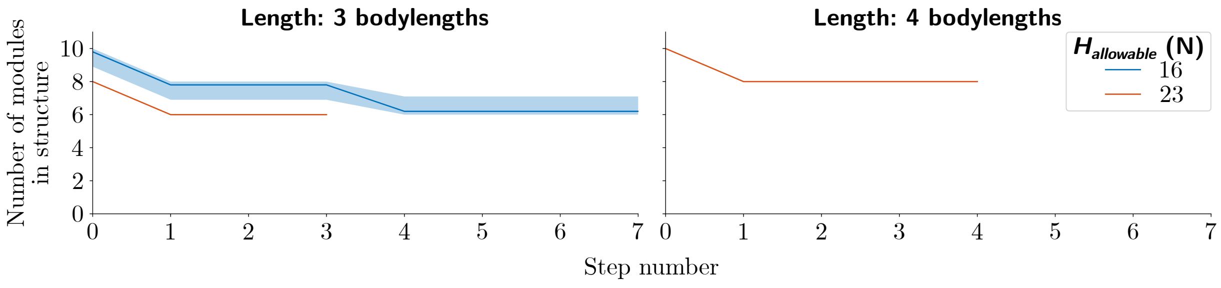
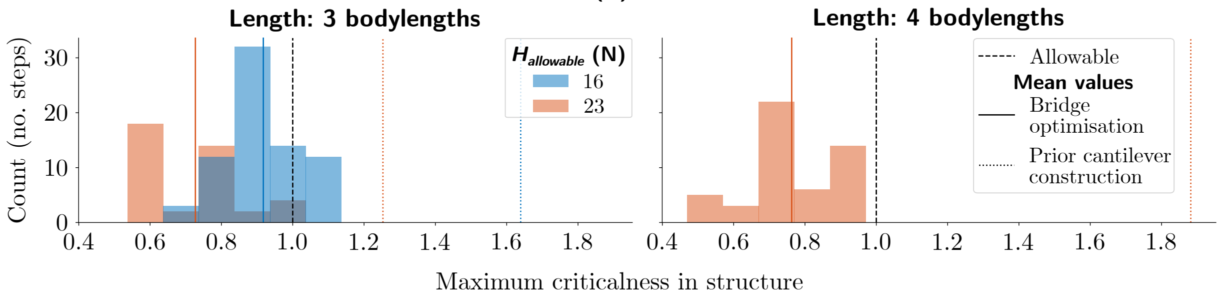


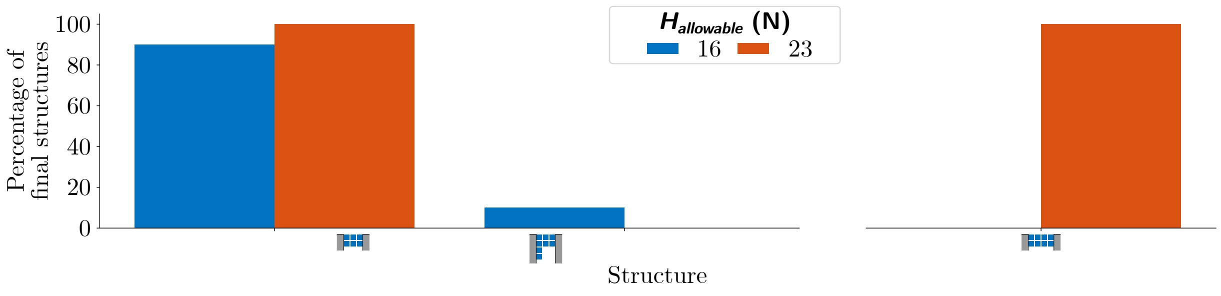
Figure 6.25. Frames from an example bridge optimisation sequence with a single active module for $H_{allowable} = 23\,\text{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



(a)



(b)



(c)

Figure 6.26. Results from the bridge optimisation algorithm with the active metamodule. **(a)** The number of modules in the structure as the trial progresses. **(b)** A histogram showing the number of steps in which the maximum criticalness measured by any strain gauge in the structure was as given. **(c)** The percentage of trials in which different final structures are built. Lines in **(a)** show mean values with the area between the 5 % and 95 % quantiles shaded, and the histogram bin width in **(b)** is 0.1.

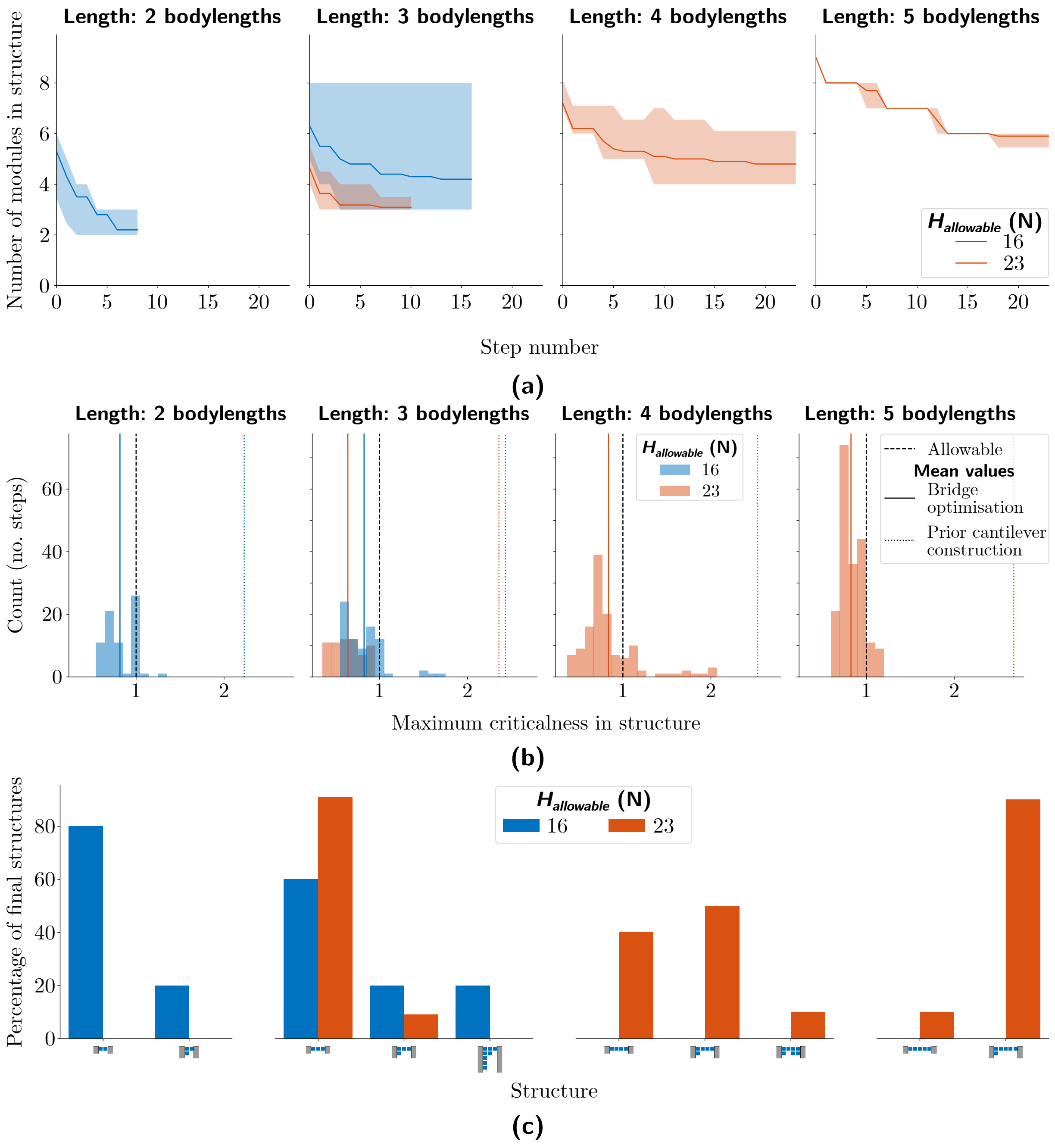


Figure 6.27. Results from the bridge optimisation algorithm with the single active module. **(a)** The number of modules in the structure as the trial progresses. **(b)** A histogram showing the number of steps in which the maximum criticalness measured by any strain gauge in the structure was as given. **(c)** The percentage of trials in which different final structures are built. Lines in **(a)** show mean values with the area between the 5 % and 95 % quantiles shaded, and the histogram bin width in **(b)** is 0.1.

nificant strength to the structure. The histograms also reveal that the mean maximum γ was lower for structures constructed with a higher $H_{allowable}$ when compared against the results for the same active agent and length.

All trials with the active metamodule for $H_{allowable} = 23\,\mathrm{N}$ resulted in the minimum number of modules required to this gap, leaving two complete rows of modules between the supports (Figure 6.26c): this structure also resulted from all but one of the trials for $H_{allowable} = 16\,\mathrm{N}$. When a single active agent was used, the smallest possible bridge is a single row of modules between the two supports. This was achieved by at least one trial for all lengths and $H_{allowable}$ tested (Figure 6.27c). This shows the algorithm is regularly able to achieve the smallest possible structure. As is the case during cantilever construction, the bridge optimisation algorithm is more likely to result in slender structures for higher $H_{allowable}$.

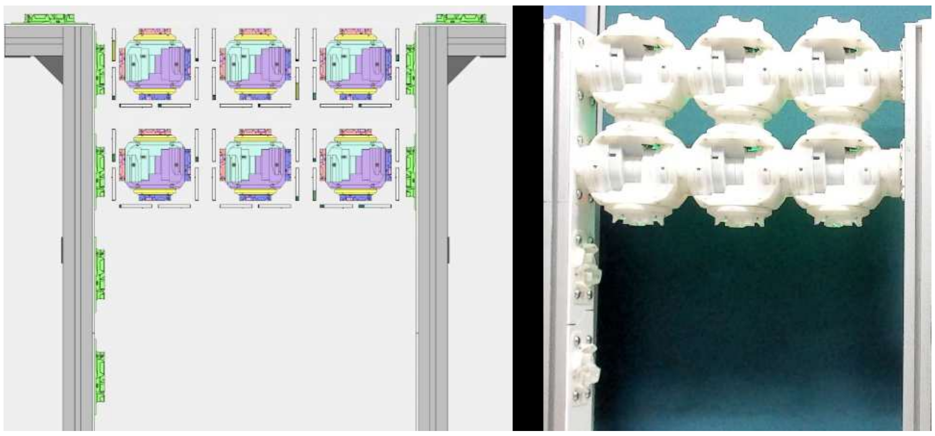
6.4.4 Bridge Deconstruction Results

The bridge deconstruction algorithm was able to successfully deconstruct the bridge without any links failing in all trials. Figure 6.28 shows how the deconstruction progresses, demonstrating a trial with an active metamodule agent that uses the optimised bridge shown in Figure 6.24 as the initial configuration. The initial bridge is stable (Figure 6.28a) but the weight of the first active agent makes the link on the left of the module in position $(1, 1)$ critical, so it places to reinforce the structure (Figure 6.28b). The next active agent measures the bridge to be stable, so flags the modules at the tip for release before placing itself (Figure 6.28c). In the next step, the removal phase is triggered, after which modules are removed from the structure columnwise (Figures 6.28d – 6.28h).

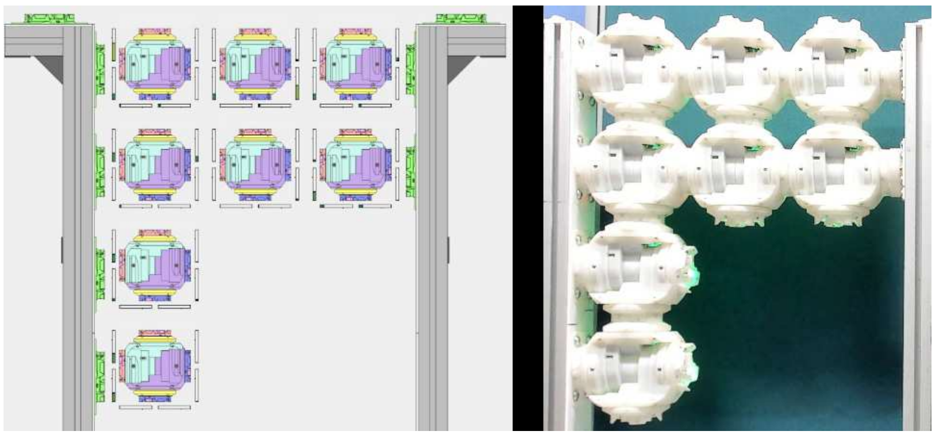
Figure 6.29 shows a trial with a single active module, initialised with the optimised bridge of Figure 6.25. The first active is added to the structure above the left support. It measures the structure to be stable and so extends the equivalent unsupported cantilever by flagging the module in position $(2, 3)$ to release next step before placing itself (Figure 6.29b). The flagged module then releases itself and becomes active. When it exits the `gathering` mode, it also believes the structure is stable, so before placing flags the module in position $(2, 4)$ to release (Figure 6.29c). Again, the next active agent measures the structure to be stable after `gathering`, but the narrow section now extends to the right support: it therefore flags the module in position $(1, 4)$ to release before placing itself (Figure 6.29d). The release of the flagged module initiates the removal phase, which continues until there are no modules remaining (Figures 6.29e – 6.29l).

Figures 6.30 & 6.31 summarise the results from all trials of the bridge deconstruction algorithm. From Figures 6.30a & 6.31a, it can be seen that the average number of modules in the structures increases to begin with as structures are reinforced, then the removal phase begins and this number drops as modules are removed. Trials with structures of the equivalent length complete faster for higher $H_{allowable}$, indicating that fewer modules need to be added before the removal phase begins. In turn, this results in fewer modules requiring removal so this phase is also completed faster. As for the bridge optimisation algorithm, all the bridge deconstruction trials with an active metamodule and $H_{allowable} = 23\,\mathrm{N}$ progressed in the same way, with the same number of agents in the structure at each step.

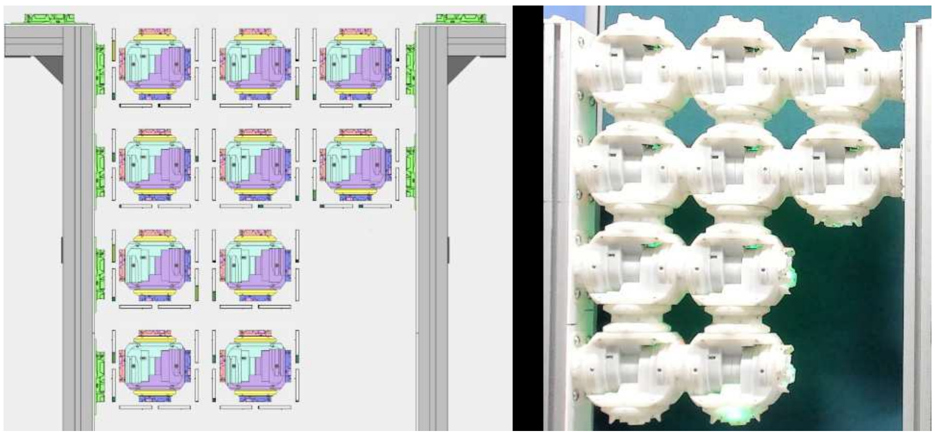
The maximum γ measured across the structure at each step during bridge deconstruc-



(a)



(b)



(c)

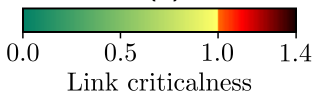
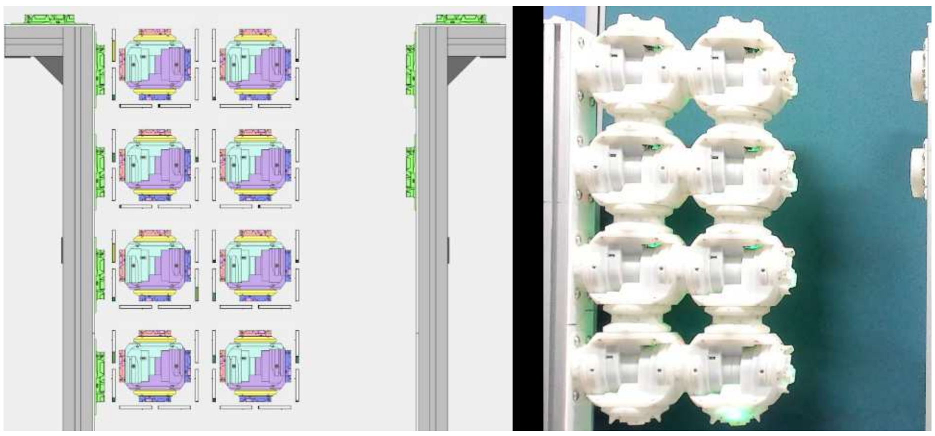
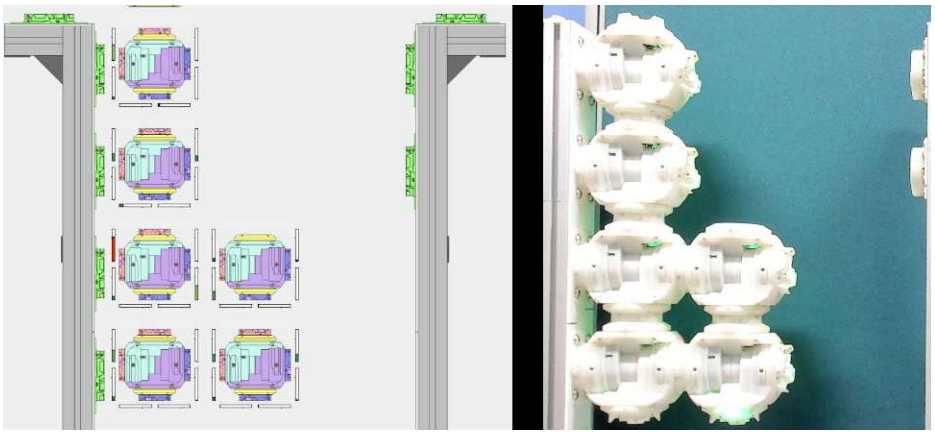
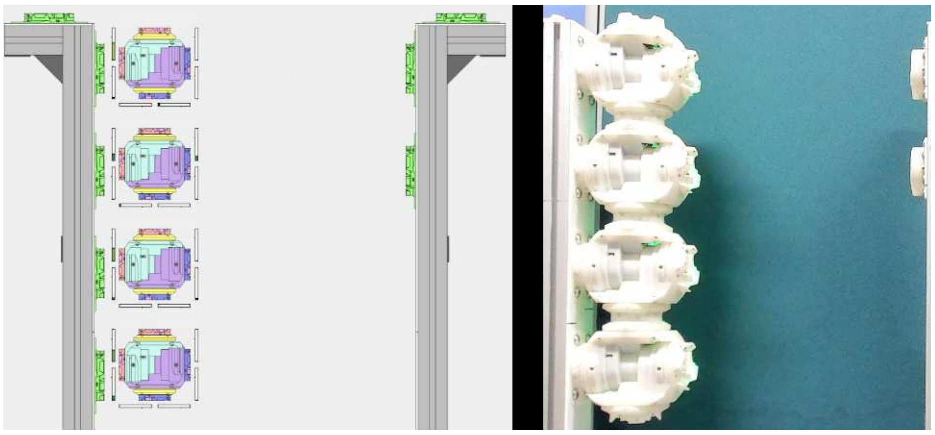


Figure 6.28. Frames from an example bridge deconstruction sequence with an active metamodule for $H_{allowable} = 16\,\text{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.


(d)


(e)


(f)

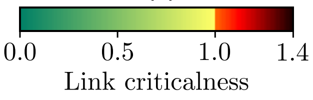
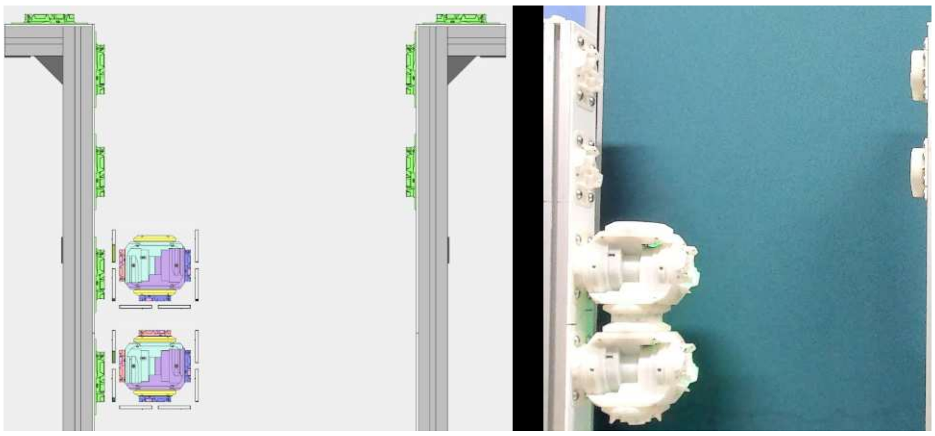
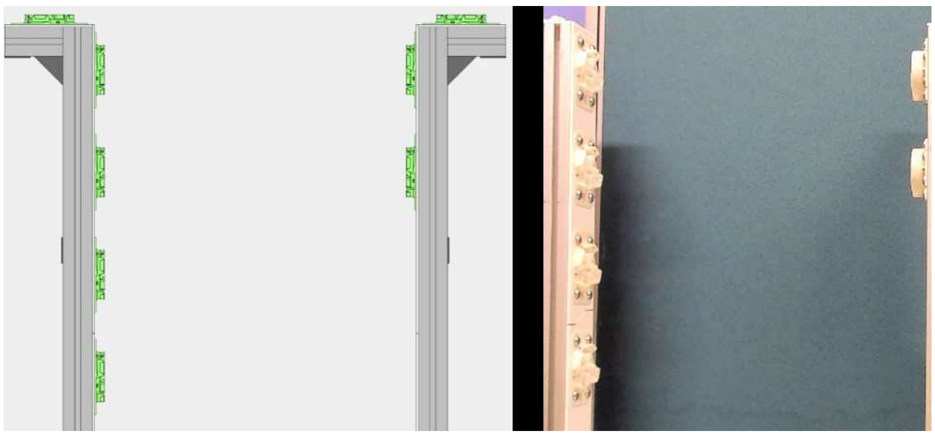


Figure 6.28. [continued] Frames from an example bridge deconstruction sequence with an active metamodule for $H_{allowable} = 16\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



(g)



(h)

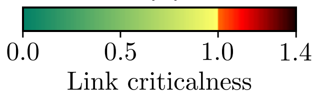
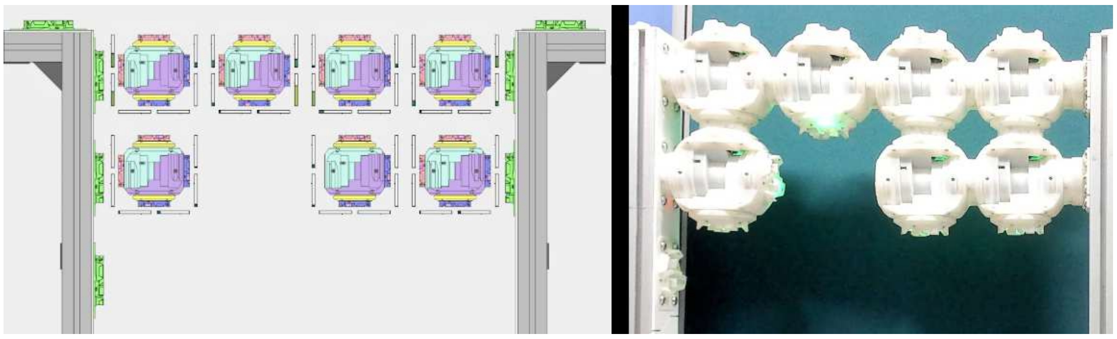
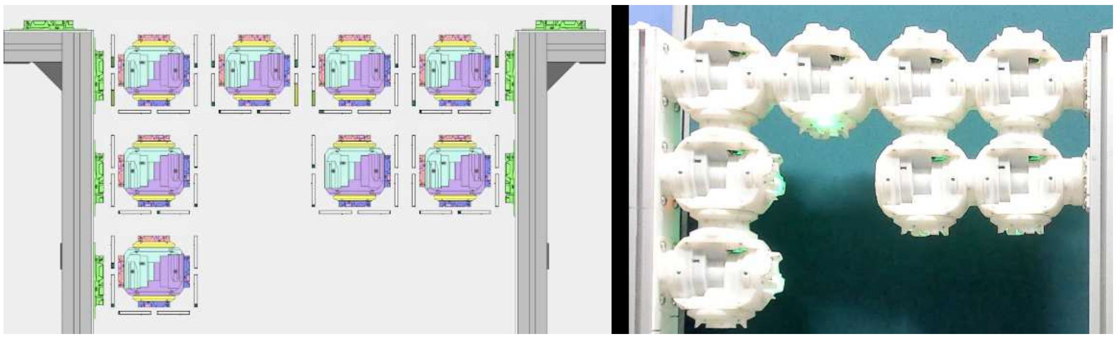


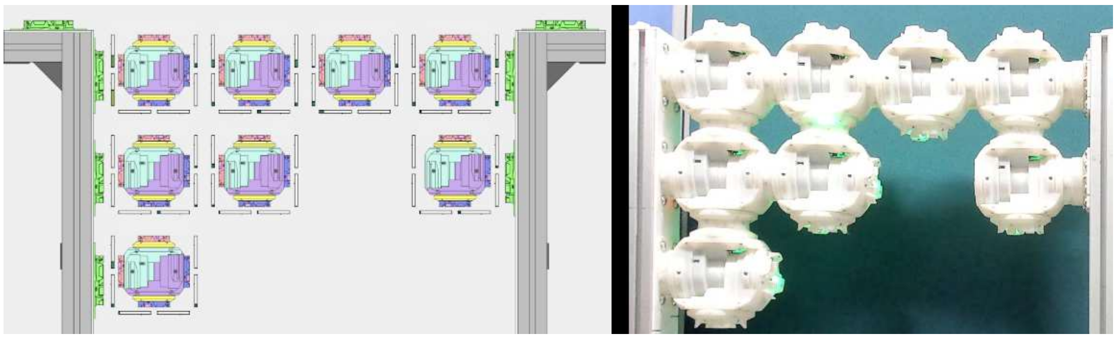
Figure 6.28. [continued] Frames from an example bridge deconstruction sequence with an active metamodule for $H_{allowable} = 16\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



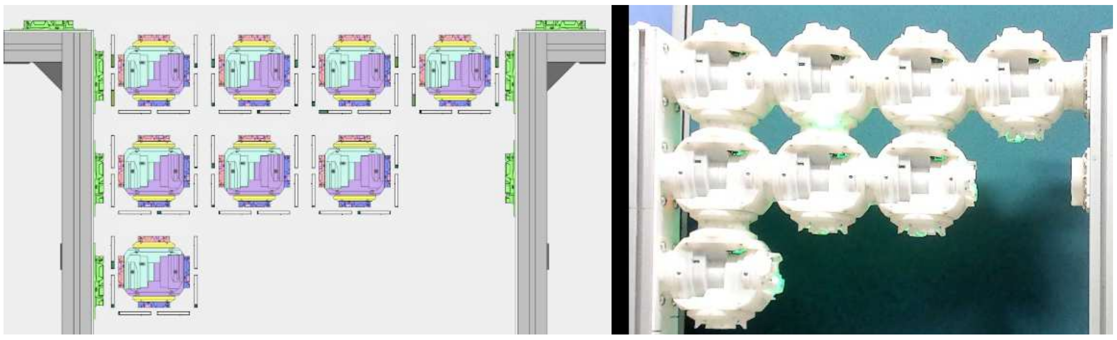
(a)



(b)



(c)



(d)

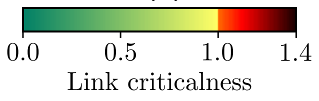
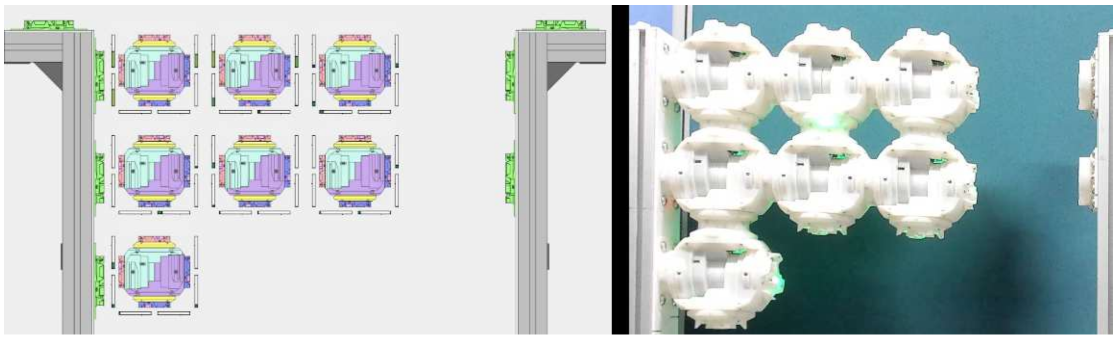
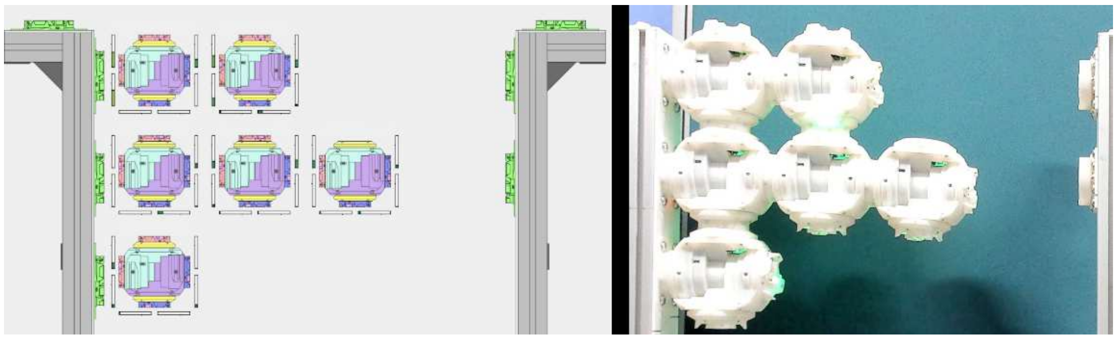
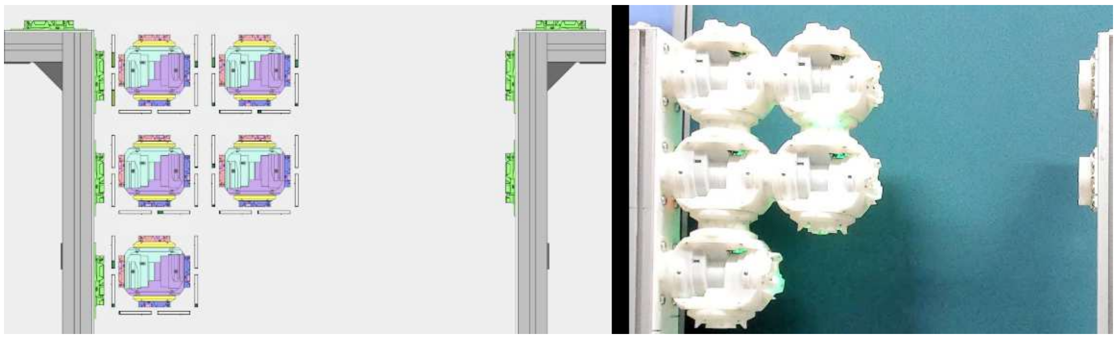
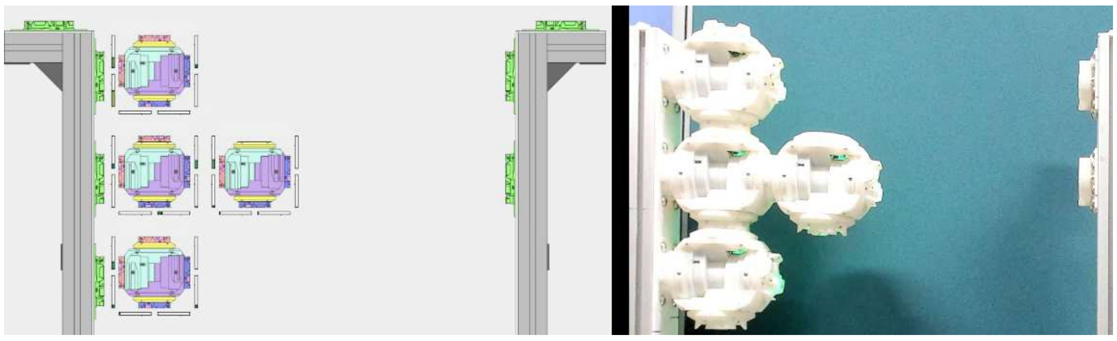


Figure 6.29. Frames from an example bridge deconstruction sequence with a single active module for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.


(e)


(f)


(g)


(h)

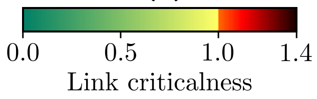
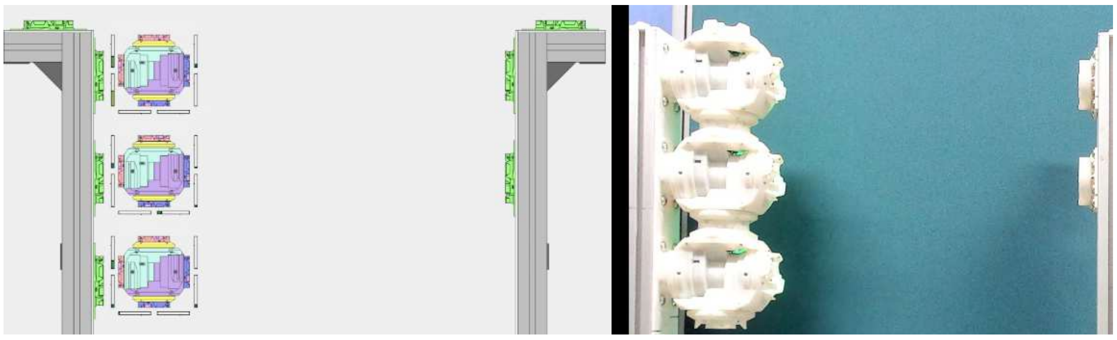
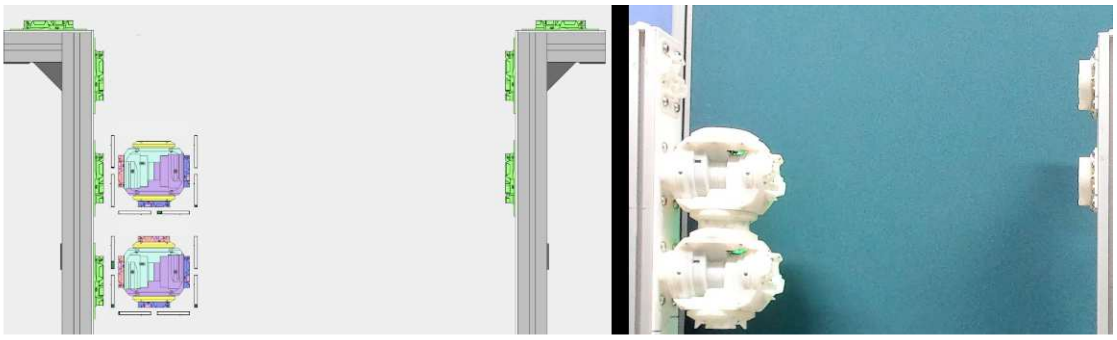
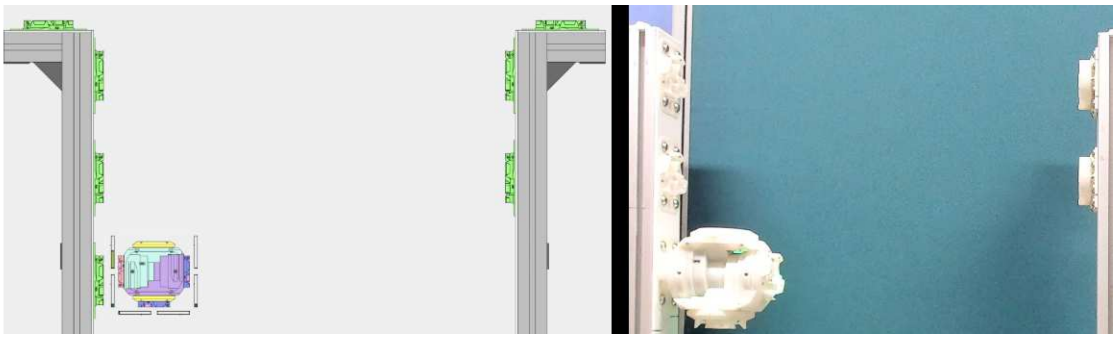
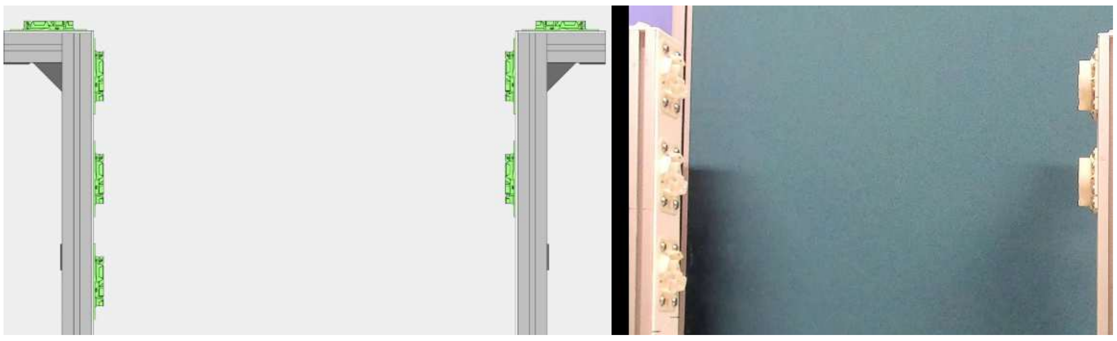


Figure 6.29. [continued] Frames from an example bridge deconstruction sequence with a single active module for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.


(i)


(j)


(k)


(l)

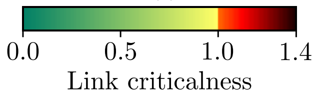
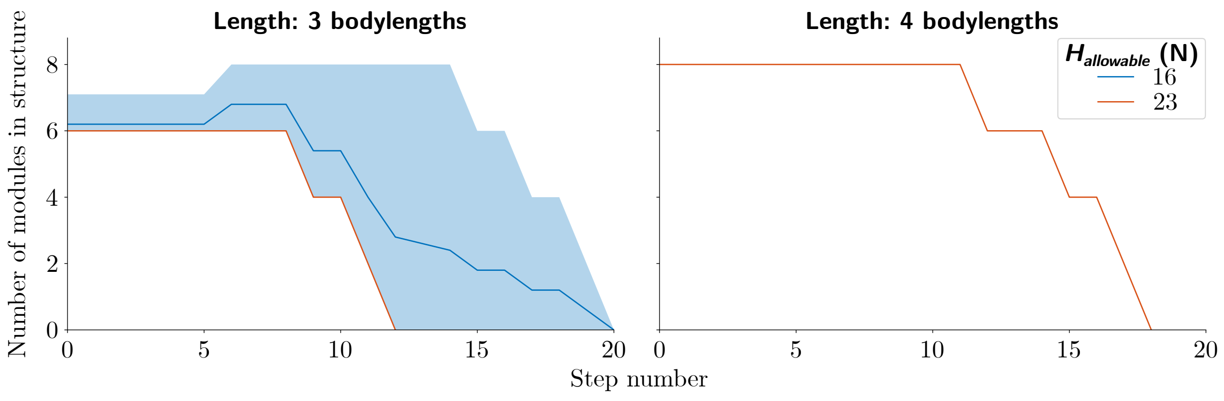
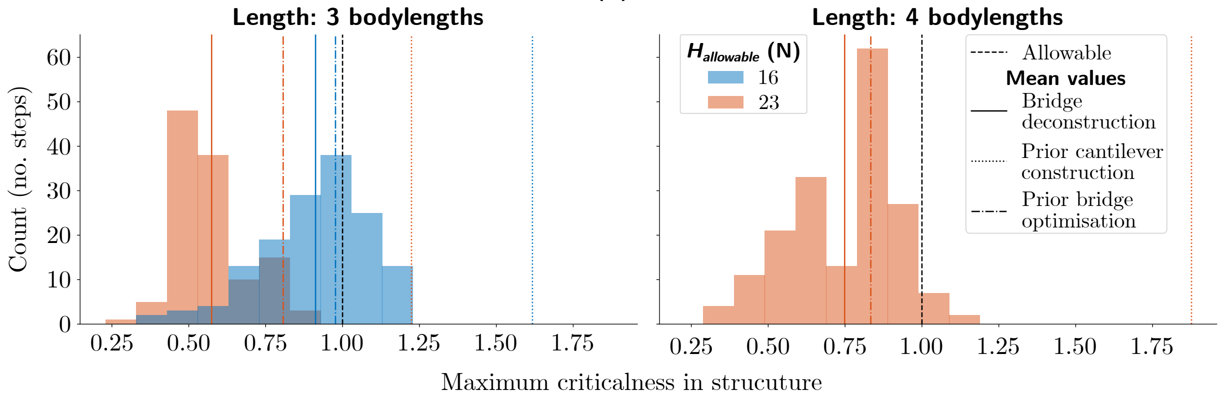


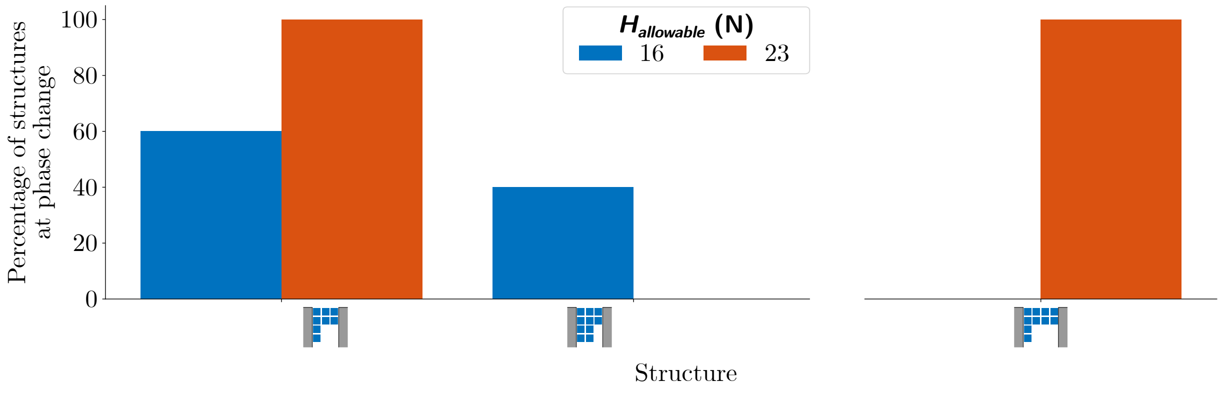
Figure 6.29. [continued] Frames from an example bridge deconstruction sequence with a single active module for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure on the right, with plots showing the measured criticalness in each strain gauge.



(a)

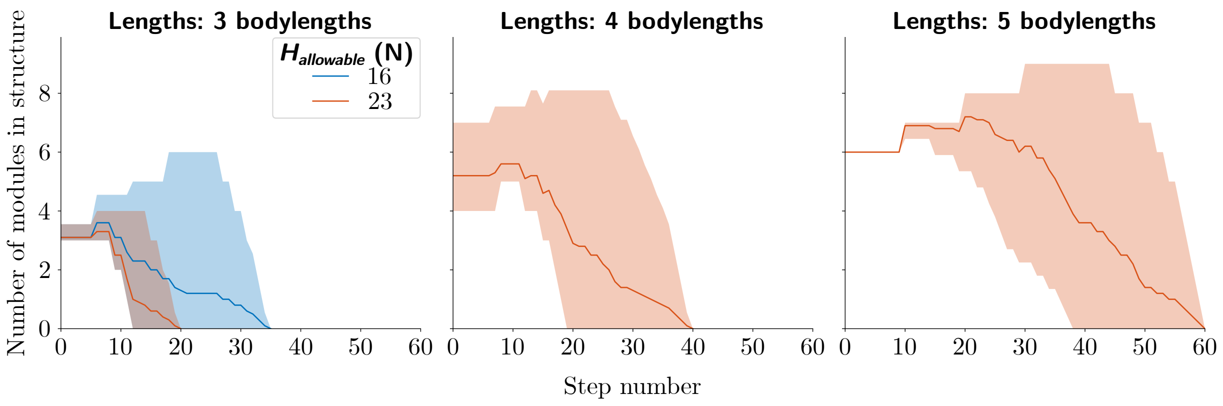


(b)

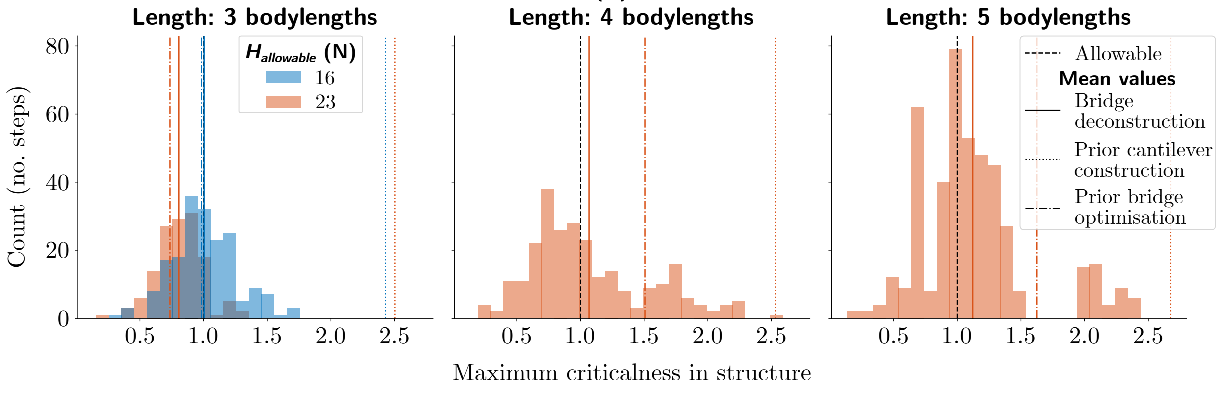


(c)

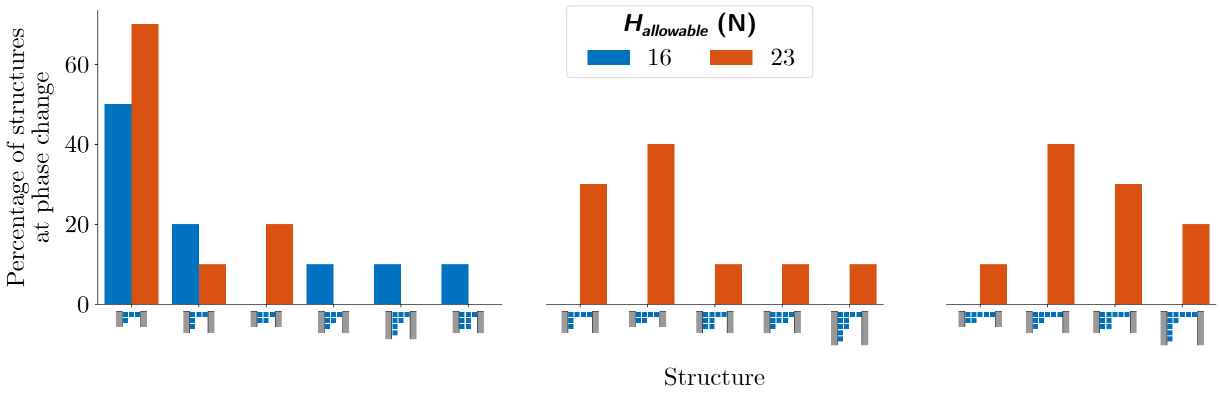
Figure 6.30. Results from the bridge deconstruction algorithm with the active metamodule. **(a)** The number of modules in the structure as the trial progresses. **(b)** A histogram showing the number of steps in which the maximum criticalness measured by any strain gauge in the structure was as given. **(c)** The percentage of trials in which different structures are built before the removal phase begins. Lines in **(a)** show mean values with the area between the 5 % and 95 % quantiles shaded, and the histogram bin width in **(b)** is 0.1.



(a)



(b)



(c)

Figure 6.31. Results from the bridge deconstruction algorithm with the single active module. **(a)** The number of modules in the structure as the trial progresses. **(b)** A histogram showing the number of steps in which the maximum criticalness measured by any strain gauge in the structure was as given. **(c)** The percentage of trials in which different structures are built before the removal phase begins. Lines in **(a)** show mean values with the area between the 5 % and 95 % quantiles shaded, and the histogram bin width in **(b)** is 0.1.

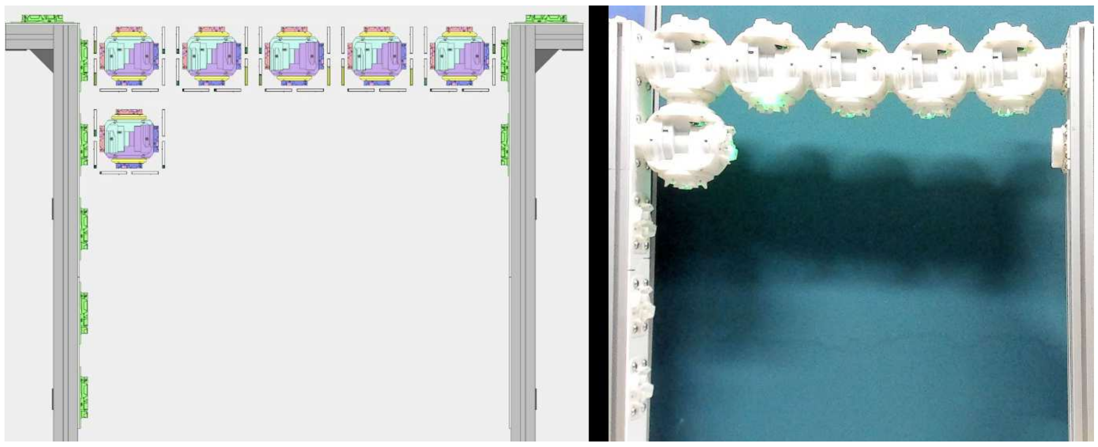
tion was generally lower than during the previous construction stages (Figures 6.30b & 6.31b). This is partly due to the high number of steps during the removal phase, in which structures are short and thick, so will naturally have a low γ. However, it can also be seen by comparing the equivalent graphs for the prior construction stages that the maximum γ during deconstruction was generally greater than during bridge optimisation, and of similar magnitude to during cantilever construction. The average γ for structures of the same length constructed by the same kind of active agent was lower when a higher $H_{allowable}$ was used, reflecting the same trend as was observed during bridge optimisation. It should also be noted that the strain gauges were only recorded when the active agent was moving around the structure: the force as soon as the right support was released is therefore excluded from these results, as it is assumed that the active agent instantaneously moves to position $\{0, L-1\}$ upon release.

The proportion of trials in which different structures were built before the removal phase begins are shown in Figures 6.30c & 6.31c. A higher $H_{allowable}$ made it more likely that structures of a given length comprised fewer agents when the removal phase began, making them more slender. This is a similar trend to the other algorithms, where higher $H_{allowable}$ allows fewer agents to be incorporated into structures of the same length.

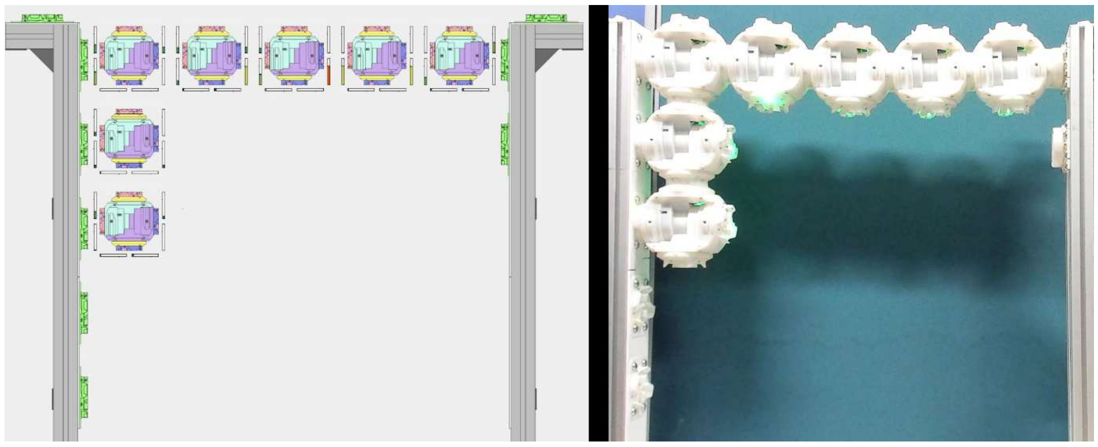
Of the 70 trials that were performed, there were two where the limited number of modules manufactured necessitated the user modifying the self-assembly sequence produced by the algorithm. This sequence was the same for both trials, and occurred for $H_{allowable} = 23\,\text{N}$ with $L = 5$ bodylengths. It is shown in Figure 6.32. The active agent repeatedly chooses column 1 to reinforce, despite this having little effect on the stability of the structure as the critical link is between columns 3 and 4 (Figures 6.32a – 6.32c). When the algorithm chooses to place the ninth module in position $(5, 1)$, the user places it in $(2, 2)$ instead (Figure 6.32d). This makes the structure stable, so the next active agent flags the module by the right support to be released in the next step. It then places in position $(5, 1)$ and the module at the tip is released (Figure 6.32e). If this change was not made, modules would have been placed in these locations in the opposite order: since the fixed support on the left only allows five rows to be created, the only remaining valid placement location with a module in $(5, 1)$ is $(2, 2)$. Placing in this order would also result in a stable structure, but it would contain all ten modules manufactured. There would therefore be no module remaining to act as the active module in the next step: this is required as the structure would not be stable when the last active agent placed itself, so the module at the tip would not be released in the next step.

6.4.5 Summary of Results

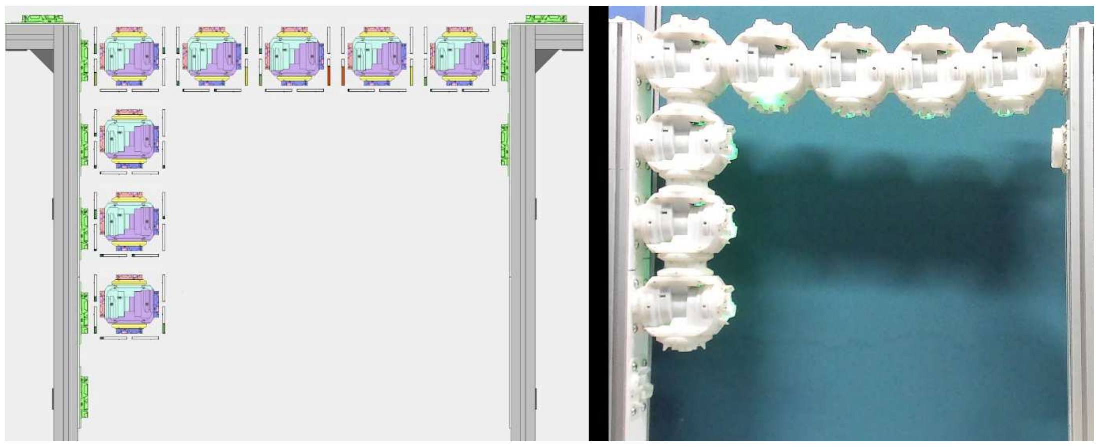
The real-world performance of the algorithms is summarised in Figure 6.33. Firstly, the number of modules used during construction, optimisation, and deconstruction of a bridge of a given length is shown in Figure 6.33a. The bridge optimisation algorithm considerably reduces the number of modules in structures of each length, $H_{allowable}$, and type of active agent tested. Agents are added to these optimised bridges during bridge deconstruction, which typically requires slightly fewer modules than during construction of a cantilever of this length before the removal phase can begin. Furthermore, these plots show how longer structures built under the same conditions require more modules, and that those built with a higher $H_{allowable}$ require fewer modules for the equivalent length.



(a)



(b)



(c)

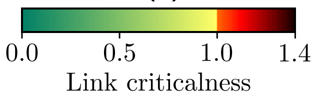
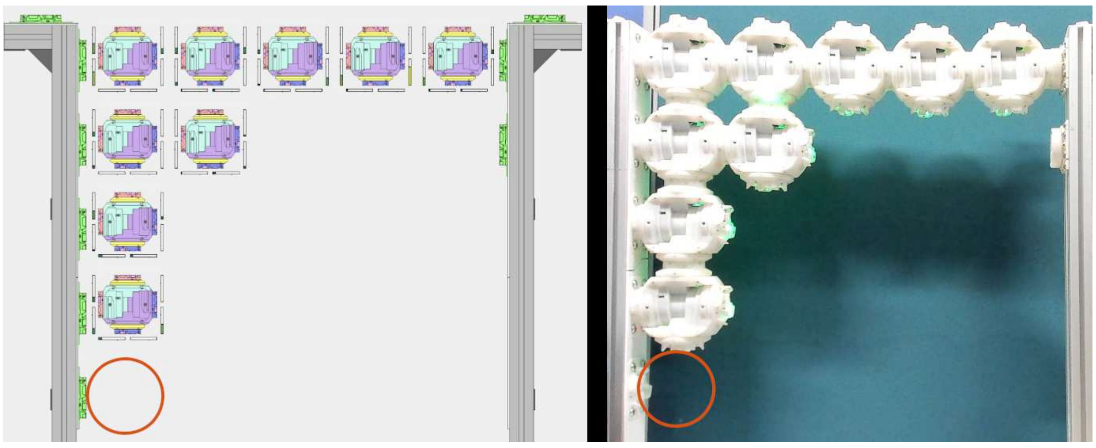
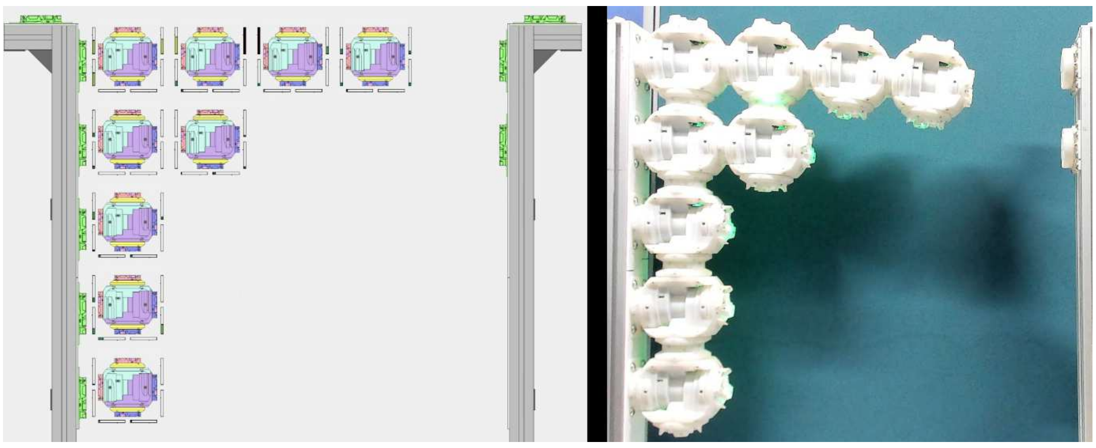


Figure 6.32. The sequence of configurations resulting from the bridge deconstruction algorithm that required the user to modify the self-assembly sequence. This occurred for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure, with plots showing the measured criticalness in each strain gauge. In **(d)**, the circled position was chosen to place the next module, but this was changed to the location pictured to allow the algorithm to complete with the ten modules manufactured.



(d)



(e)

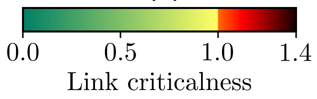
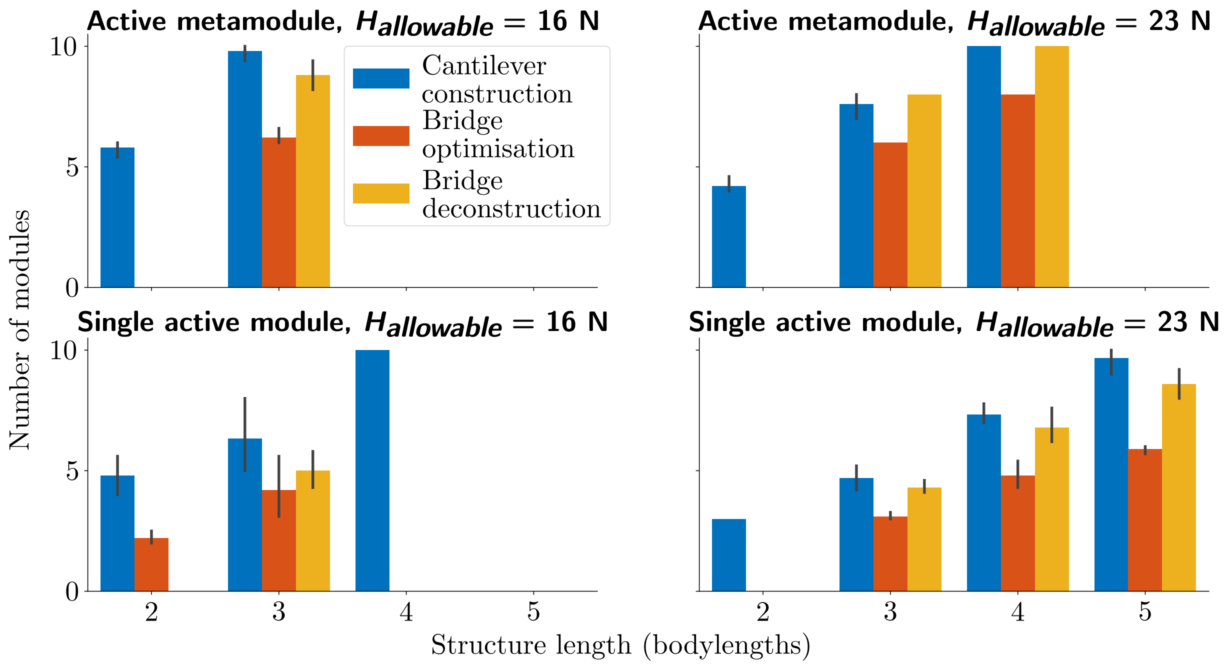
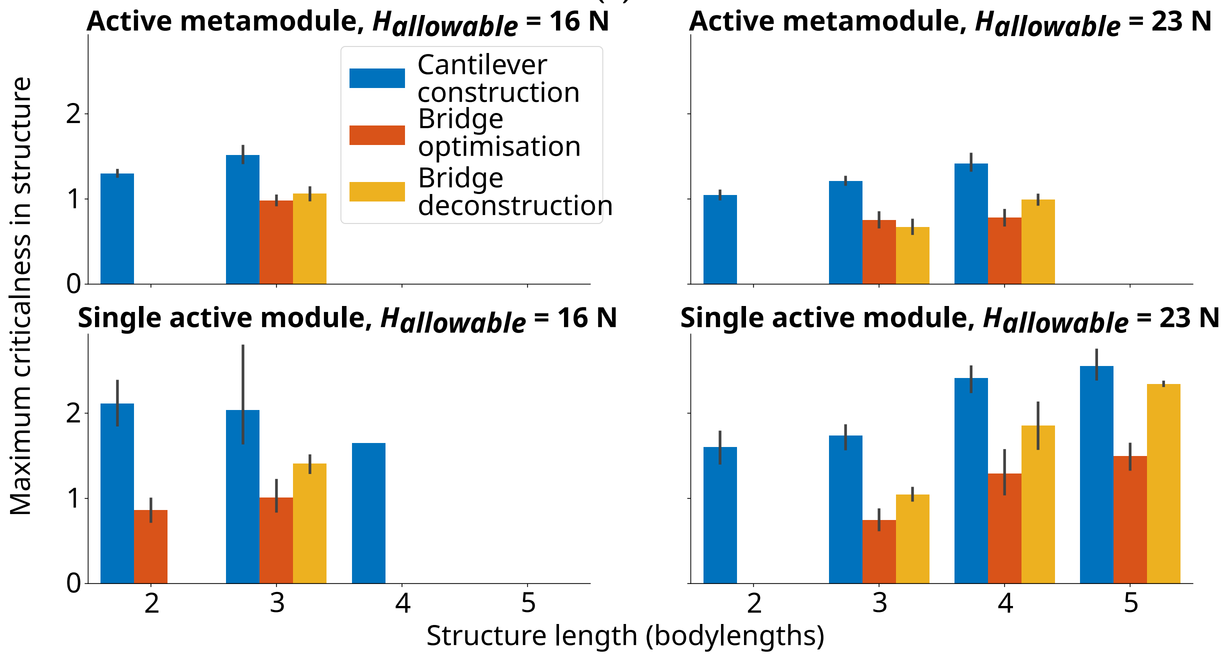


Figure 6.32. [continued] The sequence of configurations resulting from the bridge deconstruction algorithm that required the user to modify the self-assembly sequence. This occurred for $H_{allowable} = 23\,\mathrm{N}$. The left of each image shows the digital twin of the structure, with plots showing the measured criticalness in each strain gauge. In **(d)**, the circled position was chosen to place the next module, but this was changed to the location pictured to allow the algorithm to complete with the ten modules manufactured.



(a)



(b)

Figure 6.33. Summaries of the results from the real-world trials. **(a)** Comparison of the average number of modules required to construct a cantilever of a given length, the average number of modules remaining in structures after bridge optimisation, and the average number of modules required immediately before the removal phase begins during bridge deconstruction. **(b)** The maximum criticalness in the structure during cantilever construction up to the given length, and throughout bridge optimisation and deconstruction of a structure of the given length. Error bars show the 95 % confidence intervals.

Finally, using an active metamodule led to structures with more modules for the same length and $H_{allowable}$ compared to a single active module.

The maximum γ during these construction stages is shown in Figure 6.33b. The highest γ for a structure of a given length occurs during the initial cantilever construction. The forces within links are lower during bridge optimisation as the structure is supported from both ends. During bridge deconstruction, the structure is also similarly supported during the removal phase, and during the reconfiguration phase γ is reduced due to the short and thick structures that result. It is possible that a high γ would occur immediately upon the right support being released, but due to the assumed instantaneous movement of the agents when the tip is released, these values are not recorded.

6.5 Summary

This chapter has verified the self-assembly algorithms developed in the previous chapters in the real world. To perform this work, the existing HyMod platform was extensively modified to build a robotic system with the necessary capabilities. Updated HiGen connectors were first designed which incorporate strain gauges within their docking hooks to obtain the force information which the algorithms rely upon. A tensometer was used to measure the breaking strength of these connectors and to calibrate the strain gauges to give measurements in Newtons. The connectors were incorporated into two new designs of HyMod module: a passive module to build large structures from, and an active metamodule to demonstrate how a fully-autonomous system would operate. The design of the active metamodule shows how two modules can walk with a flipping gait, but modifications to reduce the backlash in the body joint and HiGen connectors are needed before fully autonomous reconfiguration can be achieved.

Minor modifications were made to the cantilever construction, bridge optimisation, and bridge deconstruction algorithms to deploy them on the newly-developed hardware. They were tested with two values of $H_{allowable}$ to observe how changing the connector strength affects the operation of the algorithms. The cantilever construction algorithm was found to be able to build cantilevers up to 5 bodylengths long without any failures in the links. The bridge optimisation algorithm could successfully reduce the number of agents in the bridge, regularly obtaining structures consisting of the minimum number of agents required to span between the two sides. The bridge deconstruction algorithm effectively deconstructed the bridge, requiring on average fewer modules than in the original cantilever. A higher $H_{allowable}$ led to longer cantilevers being constructed more quickly, fewer agents remaining in the bridges after optimisation, and fewer modules being required in the structures before the removal phase of the bridge deconstruction algorithm begins. The highest forces in the structure typically occurred during the initial cantilever construction, as opposed to during bridge optimisation and deconstruction. These are the same trends observed when the algorithms were validated in simulation in previous chapters. This demonstrates a good transfer of the force-aware self-assembly principles from simulation to the real world, despite real-world factors associated with the calibration of the strain gauges introducing noise into the force measurements.

Trials were performed with an active metamodule to demonstrate how a fully autonomous system could achieve self-assembly, and with a single active module to allow the construction of higher-resolution structures. It was found that a single active module could produce longer stable structures for a given number of agents, as they can be placed in more suitable positions than when an active metamodule is used. The effect of changing $H_{allowable}$ and the structure length has similar effects in both cases: longer structures and those built with a lower $H_{allowable}$ require more modules.

Chapter 7

Conclusions

This thesis explored the benefits of using a force-aware approach to control self-assembling modular robots. The specific task investigated was how a bridge between two vertical fixed support structures could be self-assembled by such robots to enable other agents to travel across a gap in the terrain. This is challenging to achieve without forces between agents becoming dangerously high such that they might cause the structure to collapse. A force-aware approach is therefore highly suited to this scenario. To examine this scenario, a simulation environment was first developed, in which agents reside in a square 2D grid. The configurations of agents are approximated as a truss in order to calculate an equivalent moment and axial force in their links to determine whether structures are stable or not.

Several algorithms were developed to show how force-aware methods could be applied to this scenario, each considering a different stage in the lifecycle of the bridge. In each algorithm, agents responded to measurements of force within the structure to determine what course of action they should take: whether they should add themselves to the bridge, and if so where, or whether they can be safely removed to complete other tasks. This approach means that the agents design the structure as it is built, instead of requiring a human to design separate structures for each different scenario the robots could be presented with. The algorithms operate in a distributed manner across each agent, so no centralised controller is required.

The first stage in self-assembling the bridge was considered in Chapter 3: the construction of a cantilever extending from a single fixed support. Two novel distributed force-aware self-assembly algorithms were developed to accomplish this. The initial sequential algorithm allowed for a single active agent at a time to move around the structure and decide where to place itself. Two variants of this algorithm were shown, one in which the agents coordinate to inform active agents of the maximum moment and axial force in each column of the structure, and another in which active agents only learn of these values for links belonging to agents on the perimeter of the structure. Optimal cantilevers were calculated offline through exhaustive search to use as a baseline to compare the performance of the self-assembly algorithm against. It was found that both algorithms are able to build cantilevers that are close to the optimal length for a given number of agents while keeping the moment and axial force within links in the structure within safe limits. There was little difference between the performance of the two variants, indicating that the increased coordination and communication required to give the active agent additional information about the force distribution within the structure does not yield discernible benefits.

Based on these results, a second cantilever construction algorithm was developed in which multiple active agents are allowed at once. This algorithm gathers and uses force information in the same manner as the local variant of the sequential algorithm, as this requires a lesser degree of communication than the message-passing variant and gave similar results in the earlier trials. The parallel algorithm was found to build structures that are similarly long to the sequential algorithm, but this construction occurs much quicker. It was also observed that both the sequential and parallel algorithms produced self-assembly sequences in which the maximum moment and axial force within the structure was similar.

While following either cantilever construction algorithm, the structure will eventually reach the other side of the gap. Chapter 4 considered the next stage in the self-assembly. Now the structure is supported at both ends, it is possible to remove agents from the structure while maintaining stability so that they can complete other tasks. The bridge optimisation algorithm was developed to enable this behaviour, which is also force-aware and runs in a distributed manner across each agent. Following this algorithm, agents on the lower perimeter of the structure release themselves when they believe they are not usefully contributing to the structural stability of the bridge. They then gather information about the state of the structure, and either remove themselves from it or replace somewhere else to provide reinforcement in a more suitable location.

Optimal bridges were also calculated offline to compare the performance of the bridge optimisation algorithm against. It was found that the algorithm could reduce the number of agents in the bridge to nearly the optimal amount. When fewer agents are active simultaneously, they are able to receive more accurate force information and thus place in more suitable locations. This means the algorithm is able to produce structures with closer to the optimal number of agents. The maximum moment and axial force in links within the structure while running this algorithm was usually less than during the self-assembly of the cantilever used as a starting point, which was generated from the parallel cantilever construction algorithm.

Eventually, the bridges will no longer be required. In Chapter 5 the bridge deconstruction algorithm was developed to enable the bridge to be dismantled in a safe and distributed manner, once again incorporating local force information in the decision-making processes. The algorithm begins by reinforcing the structure by adding new agents and repositioning existing ones; when it is believed the structure is able to safely support itself from only one side, all agents are removed columnwise. The algorithm was tested for a range of structures resulting from the bridge optimisation algorithm, and was found to successfully deconstruct all of them. In most cases, a similar number of agents to the original cantilever was required before the removal phase could begin. It was also found that the maximum moment and axial force within the structure during deconstruction was typically similar to that during the prior cantilever construction stage.

In Chapter 6, a new robotic platform was developed to enable these algorithms to be verified in real life, based on the design of the existing HyMod robots [25]. The modules were extensively modified to allow these algorithms to be deployed on them. The most important modification is that the HiGen connectors these modules incorporate were enhanced with strain gauges to enable them to sense the force within links between modules. Two types of module were designed: a passive module to build the structures from and hence verify the force-aware algorithms in real-life, and an active metamodule to demonstrate how two modules could move around the structure in a fully autonomous system.

The force-aware HiGen connectors were able to reliably measure the force in their individual docking hooks through strain gauges. However, reliably converting these readings to separate measurements of moment and axial force was not possible due to backlash in the connectors. The passive modules were able to successfully build a wide range of large structures, indicating the potential the platform has in realising these algorithms in real life. The active metamodule could autonomously turn all of its joints through their full range of motion in a variety of orientations, which demonstrates how the design can achieve the necessary motions to autonomously enact these algorithms in a future version of the design. However, due to backlash in the body joints it was not possible to reliably walk along other modules without human intervention.

The algorithms were found to translate well from simulation to the real world. They were tested for two kinds of active module: an active metamodule to demonstrate how a fully-autonomous system would function, and a single active module to evaluate the performance for a larger number of placement decisions. Similar trends were observed to simulation, with higher allowable forces resulting in longer cantilevers for a given number of agents, stable bridges of equivalent length containing fewer agents, and fewer agents required to add to the structure during deconstruction before agents could be safely removed. The highest measured forces in the structure occurred during the initial cantilever construction. Using a single active module allowed for longer structures to be built with the same number of modules, as they could be placed in more suitable locations due to the higher structural resolution they offered.

7.1 Future Work

While this thesis presents a thorough study of how force-aware self-assembly can be applied to build bridges from modular robots, there are still many avenues in which this work could be extended. One example is in how columns are chosen to reinforce based on the received force information. The proposed stochastic method is shown to produce structures that are close to the optimum, and is likely to be robust to reasonable levels of sensor noise, as was briefly mentioned for the current hardware implementation. However, agents occasionally place in locations that a human observer can clearly see are not the most suitable, resulting in suboptimal structures and increased self-assembly time. Future work could investigate how this stochastic method could be refined to improve the choice of placement locations.

Other future work could consider how the framework could be extended to similar problems. One such extension could be the construction of structures between platforms of different heights, instead of building horizontal bridges. This would present additional challenges in correctly supporting the structure when building more complex structures. Alternatively, further research could consider how to adapt the bridge to support additional loads. If the agents knew they were to build a bridge capable of supporting loads in addition to themselves, for example to transport objects across the structure, they could build stronger bridges, based on the same force-aware methodology. This could either take place during the bridge optimisation stage, or a further stage could be introduced where agents are added back into the structure when it is known that additional loads should be supported. The principle of superposition could possibly be used again in this scenario to model what the forces would be under this additional loading, or methods of calculating suitable allowable forces to support given loads could be devised.

This work considered a 2D grid of square agents, but similar force-aware methods could enable the construction of bridges in different environments. Extending the algorithms into 3D would be a very interesting avenue of future research, which would make the structures more suitable to real-world scenarios. It would also be of great interest to consider other modular robotic topologies, such as freeform systems, to evaluate the benefits of force-aware control in this upcoming research area.

The algorithms were developed in simulation, and any future improvements are also likely to be investigated using simulated robots due to the current cost and availability of self-assembling modular robotic platforms. In order to improve the confidence in the simulated algorithms, the simulation environment could also be improved. In particular, future studies could incorporate the deformation of the structure into the simulations, and develop algorithms which reduce the deflection of the structure under loading, as well as limiting forces within the links between agents. Another possible improvement to the simulator would be to model the motions of specific platforms, such as HyMod, to investigate how the dynamic forces as agents more around the structure affect the measurements of force. It is possible that agents could exert significant forces on the structure as they actuate, so the transfer of the algorithms from simulation to the real world could benefit from the modelling of such forces.

The new HyMod modules developed to demonstrate the algorithms in real life were suitable for initial proof-of-concept trials, but not capable of creating a fully-autonomous system. The design could be improved in certain areas, in particular by reducing backlash in the body joints and HiGen connectors, and manufacturing certain critical components from stronger materials. This would enable accurate and repeatable motions to be reliably achieved. It would therefore be possible to implement the algorithms without the need for a human operator to move the modules around the structure. This would further validate the algorithms, and truly demonstrate their potential in autonomous systems.

This thesis hopes to inspire future researchers to investigate how force-aware self-assembly can be applied to a range of problems. The self-assembly of bridges is an example where these methods are particularly suited, but the concept can be applied to a broad spectrum of problems. Force-aware modular manipulators could reconfigure to move loads of different weights. Self-assembling modular robots configured as walkers could modify their arrangement based on measured forces to allow them to carry objects across a variety of different terrains. These examples and countless others demonstrate the potential of force-aware self-assemby, and exploring any of them would bring autonomous self-assembling modular robots one step closer to realising their huge potential in solving real-world problems.

Bibliography

[1] L. E. Parker, D. Rus, and G. S. Sukhatme, “Multiple Mobile Robot Systems,” in *Springer Handbook of Robotics*, ser. Springer Handbooks, B. Siciliano and O. Khatib, Eds., Cham: Springer International Publishing, 2016, pp. 1335–1384.

[2] K. Stoy, D. Brandt, and D. Christensen, *Self-Reconfigurable Robots: An Introduction*. Cambridge: MIT press, 2010.

[3] J. Seo, J. Paik, and M. Yim, “Modular Reconfigurable Robotics,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 63–88, 2019.

[4] R. Groß and M. Dorigo, “Self-Assembly at the Macroscopic Scale,” *Proceedings of the IEEE*, vol. 96, no. 9, pp. 1490–1508, 2008.

[5] R. R. Murphy *et al.*, “Search and Rescue Robotics,” in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Berlin, Heidelberg: Springer, 2008, pp. 1151–1173.

[6] Y. Koren *et al.*, “Reconfigurable Manufacturing Systems,” *CIRP Annals*, vol. 48, no. 2, pp. 527–540, 1999.

[7] S. Hauser, M. Mutlu, P. A. Léziart, H. Khodr, A. Bernardino, and A. J. Ijspeert, “Roombots extended: Challenges in the next generation of self-reconfigurable modular robots and their application in adaptive and assistive furniture,” *Robotics and Autonomous Systems*, vol. 127, p. 103 467, 2020.

[8] M. Yim, K. Roufas, D. Duff, Y. Zhang, C. Eldershaw, and S. Homans, “Modular Reconfigurable Robots in Space Applications,” *Autonomous Robots*, vol. 14, no. 2, pp. 225–237, 2003.

[9] United Nations, “Transforming our world: The 2030 Agenda for Sustainable Development,” *United Nations*, 2015.

[10] J. Amorim Marques, M.-T. Lorente, and R. Groß, “Multi-robot systems research: A data-driven trend analysis,” in *2022 International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Springer International Publishing, 2022.

[11] L. Villani and J. De Schutter, “Force Control,” in *Springer Handbook of Robotics*, ser. Springer Handbooks, B. Siciliano and O. Khatib, Eds., Cham: Springer International Publishing, 2016, pp. 195–220.

[12] K. H. Petersen, N. Napp, R. Stuart-Smith, D. Rus, and M. Kovac, “A review of collective robotic construction,” *Science Robotics*, vol. 4, no. 28, 2019.

[13] J. Werfel, K. Petersen, and R. Nagpal, “Designing Collective Behavior in a Termite-Inspired Robot Construction Team,” *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[14] M. McEvoy, E. Komendera, and N. Correll, “Assembly path planning for stable robotic construction,” in *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, IEEE, 2014, pp. 1–6.

[15] P. Funes and J. Pollack, “Computer Evolution of Buildable Objects,” in *Evolutionary Design by Computers*, P. J. Bentley, Ed., vol. 1, Morgan Kaufmann, 1999, pp. 387–403.

[16] L. Brodbeck and F. Iida, “Automatic real-world assembly of machine-designed structures,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 1221–1226.

[17] N. Melenbrink, P. Kassabian, A. Menges, and J. Werfel, “Towards Force-aware Robot Collectives for On-site Construction,” in *Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, 2017, pp. 382–391.

[18] Y. Suzuki, N. Inou, H. Kimura, and M. Koseki, “Self-Reconfigurable Modular Robots Adaptively Transforming a Mechanical Structure: Algorithm for Adaptive Transformation to Load Condition,” *Journal of Robotics*, vol. 2011, e794251, 2011.

[19] C. Zhang, W. Zou, L. Ma, and Z. Wang, “Biologically inspired jumping robots: A comprehensive review,” *Robotics and Autonomous Systems*, vol. 124, p. 103 362, 2020.

[20] M. Saboia, V. Thangavelu, and N. Napp, “Autonomous multi-material construction with a heterogeneous robot team,” *Robotics and Autonomous Systems*, vol. 121, p. 103 239, 2019.

[21] V. Thangavelu, M. S. da Silva, J. Choi, and N. Napp, “Autonomous Modification of Unstructured Environments with Found Material,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 7798–7804.

[22] N. Inou, S. Fukushima, N. Shimotai, and S. Ujihashi, “Study of Group Robots Adaptively Forming a Mechanical Structure : Effect of Mechanical Properties of Cellular Robots on Structure Formation,” *JSME International Journal Series C*, vol. 43, no. 1, pp. 127–133, 2000.

[23] N. Inou, N. Shimotai, S. Fukushima, H. Ogawa, and S. Ujihashi, “Study of Group Robots Adaptively Forming a Mechanical Structure : Information-Processing Functions of the Cellular Robot Required for Transporting a Load to an Opposite Side,” *JSME International Journal Series C*, vol. 43, no. 1, pp. 134–140, 2000.

[24] P. Swissler and M. Rubenstein, “ReactiveBuild: Environment-Adaptive Self-Assembly of Amorphous Structures,” in *2021 International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Springer International Publishing, 2021, pp. 363–375.

[25] C. Parrott, T. J. Dodd, and R. Groß, “HyMod: A 3-DOF Hybrid Mobile and Self-Reconfigurable Modular Robot and its Extensions,” in *Distributed Autonomous Robotic Systems: 13th International Symposium*, Springer International Publishing, 2018, pp. 401–414.

[26] C. Parrott, T. J. Dodd, and R. Groß, "HiGen: A high-speed genderless mechanical connection mechanism with single-sided disconnect for self-reconfigurable modular robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 3926–3932.

[27] W. Cresswell, "Flocking is an effective anti-predation strategy in redshanks, Tringa totanus," *Animal Behaviour*, vol. 47, no. 2, pp. 433–442, 1994.

[28] R. Pitman and J. Durban, "Cooperative hunting behavior, prey selectivity and prey handling by pack ice killer whales (Orcinus orca), type B, in Antarctic Peninsula waters," *Marine Mammal Science*, vol. 28, pp. 16–36, 2012.

[29] P. Collen and R. Gibson, "The general ecology of beavers (Castor spp.), as related to their influence on stream ecosystems and riparian habitats, and the subsequent effects on fish – a review," *Reviews in Fish Biology and Fisheries*, vol. 10, no. 4, pp. 439–461, 2000.

[30] M. C. Mainwaring, I. R. Hartley, M. M. Lambrechts, and D. C. Deeming, "The design and function of birds' nests," *Ecology and Evolution*, vol. 4, no. 20, pp. 3909–3928, 2014.

[31] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, E. Bonabeau, and G. Theraulaz, *Self-Organization in Biological Systems.* Princeton University Press, 2003.

[32] P.-P. Grassé, "La reconstruction du nid et les coordinations inter-individuelles chez Bellicositermes natalensis et Cubitermes sp. La theorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs," *Insectes sociaux*, vol. 6, no. 1, pp. 41–80, 1959.

[33] N. J. Mlot, C. A. Tovey, and D. L. Hu, "Fire ants self-assemble into waterproof rafts to survive floods," *Proceedings of the National Academy of Sciences*, vol. 108, no. 19, pp. 7669–7673, 2011.

[34] C. Anderson, G. Theraulaz, and J.-L. Deneubourg, "Self-assemblages in insect societies," *Insectes Sociaux*, vol. 49, no. 2, pp. 99–110, 2002.

[35] S. Phonekeo, N. Mlot, D. Monaenkova, D. L. Hu, and C. Tovey, "Fire ants perpetually rebuild sinking towers," *Royal Society Open Science*, vol. 4, no. 7, p. 170 475, 2017.

[36] S. Powell and N. R. Franks, "How a few help all: Living pothole plugs speed prey delivery in the army ant Eciton burchellii," *Animal Behaviour*, vol. 73, no. 6, pp. 1067–1076, 2007.

[37] P. C. Foster, N. J. Mlot, A. Lin, and D. L. Hu, "Fire ants actively control spacing and orientation within self-assemblages," *Journal of Experimental Biology*, vol. 217, no. 12, pp. 2089–2100, 2014.

[38] A. Prayoga. "No bridge too far! Army of ants stranded in tree build their own bridge by climbing on top of each other," Daily Mail. (2014), [Online]. Available: `https://www.dailymail.co.uk/news/article-2603204/No-bridge-far-Army-ants-stranded-tree-build-bridge-climbing-other.html` (visited on Jul. 29, 2020).

[39] C. R. Reid, M. J. Lutz, S. Powell, A. B. Kao, I. D. Couzin, and S. Garnier, "Army ants dynamically adjust living bridges in response to a cost–benefit trade-off," *Proceedings of the National Academy of Sciences*, vol. 112, no. 49, pp. 15 113–15 118, 2015.

[40] M. J. Lutz, C. R. Reid, C. J. Lustri, A. B. Kao, S. Garnier, and I. D. Couzin, "Individual error correction drives responsive self-assembly of army ant scaffolds," *Proceedings of the National Academy of Sciences*, vol. 118, no. 17, e2013741118, 2021.

[41] B. Hölldobler and E. O. Wilson, "The Multiple Recruitment Systems of the African Weaver Ant Oecophylla longinoda (Latreille)(Hymenoptera: Formicidae)," *Behavioral Ecology and Sociobiology*, pp. 19–60, 1978.

[42] T. Fukuda and S. Nakagawa, "A Dynamically Reconfigurable Robotic System (Concept Of A System And Optimal Configurations)," in *IECON '87: Industrial Applications of Robotics & Machine Vision*, vol. 0856, SPIE, 1987, pp. 588–595.

[43] M. Yim *et al.*, "Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[44] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "A review on modelling, implementation, and control of snake robots," *Robotics and Autonomous Systems*, vol. 60, no. 1, pp. 29–40, 2012.

[45] J. Sastra, S. Chitta, and M. Yim, "Dynamic Rolling for a Modular Loop Robot," *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 758–773, 2009.

[46] M. Yim, B. Shirmohammadi, J. Sastra, M. Park, M. Dugan, and C. J. Taylor, "Towards robotic self-reassembly after explosion," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 2767–2772.

[47] A. Castano, A. Behar, and P. Will, "The Conro modules for reconfigurable robots," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 403–409, 2002.

[48] J. Baca, S. G. M. Hossain, P. Dasgupta, C. A. Nelson, and A. Dutta, "ModRED: Hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration," *Robotics and Autonomous Systems*, Reconfigurable Modular Robotics, vol. 62, no. 7, pp. 1002–1015, 2014.

[49] L. Pfotzer, S. Ruehl, G. Heppner, A. Roennau, and R. Dillmann, "KAIRO 3: A modular reconfigurable robot for search and rescue field missions," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, 2014, pp. 205–210.

[50] M. Yim, "New locomotion gaits," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, 2508–2514 vol.3.

[51] M. Yim, D. Duff, and K. Roufas, "PolyBot: A modular reconfigurable robot," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, 2000, 514–520 vol.1.

[52] M. Yim, Y. Zhang, K. Roufas, D. Duff, and C. Eldershaw, “Connecting and disconnecting for chain self-reconfiguration with PolyBot,” *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 442–451, 2002.

[53] M. Park, S. Chitta, A. Teichman, and M. Yim, “Automatic configuration recognition methods in modular robots,” *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 403–421, 2008.

[54] L. Pfotzer, S. Klemm, A. Roennau, J. M. Zöllner, and R. Dillmann, “Autonomous navigation for reconfigurable snake-like robots in challenging, unknown environments,” *Robotics and Autonomous Systems*, vol. 89, pp. 123–135, 2017.

[55] S. Murata, H. Kurokawa, and S. Kokaji, “Self-assembling machine,” in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, 441–448 vol.1.

[56] C. H. Belke and J. Paik, “Mori: A Modular Origami Robot,” *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 5, pp. 2153–2164, 2017.

[57] C. Parrott, “A Hybrid and Extendable Self-Reconfigurable Modular Robotic System,” Ph.D. dissertation, The University of Sheffield, 2016.

[58] D. Rus and M. Vona, “A physical implementation of the self-reconfiguring crystalline robot,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, 1726–1733 vol.2.

[59] Y. Suzuki, N. Inou, M. Koseki, and H. Kimura, “Reconfigurable group robots adaptively transforming a mechanical structure - Extended criteria for load-adaptive transformations -,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 877–882.

[60] J. W. Romanishin, K. Gilpin, S. Claici, and D. Rus, “3D M-Blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1925–1932.

[61] A. Spröwitz *et al.*, “Roombots: Reconfigurable Robots for Adaptive Furniture,” *IEEE Computational Intelligence Magazine*, vol. 5, no. 3, pp. 20–32, 2010.

[62] E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund, “Design of the ATRON lattice-based self-reconfigurable robot,” *Autonomous Robots*, vol. 21, no. 2, pp. 165–183, 2006.

[63] K. Gilpin, A. Knaian, and D. Rus, “Robot pebbles: One centimeter modules for programmable matter through self-disassembly,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2485–2492.

[64] D. Rus and M. Vona, “Crystalline Robots: Self-Reconfiguration with Compressible Unit Modules,” *Autonomous Robots*, vol. 10, no. 1, pp. 107–124, 2001.

[65] J. Suh, S. Homans, and M. Yim, “Telecubes: Mechanical design of a module for self-reconfigurable robotics,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 4, 2002, 4095–4101 vol.4.

[66] A. Pamecha, C.-J. Chiang, D. Stein, and G. Chirikjian, “Design and implementation of metamorphic robots,” in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 97577, American Society of Mechanical Engineers, 1996, V02AT02A013.

[67] M. Koseki, K. Minami, and N. Inou, “Cellular Robots Forming a Mechanical Structure (Evaluation of structural formation and hardware design of ”CHOBIE II”),” in *Distributed Autonomous Robotic Systems (DARS)*, Tokyo: Springer Japan, 2004, pp. 131–140.

[68] B. K. An, “Em-cube: Cube-shaped, self-reconfigurable robots sliding on structure surfaces,” in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 3149–3155.

[69] S. Hauser, M. Mutlu, and A. J. Ijspeert, “Kubits: Solid-State Self-Reconfiguration With Programmable Magnets,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6443–6450, 2020.

[70] M. Nisser, L. Cheng, Y. Makaram, R. Suzuki, and S. Mueller, “ElectroVoxel: Electromagnetically Actuated Pivoting for Scalable Modular Self-Reconfigurable Robots,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4254–4260.

[71] B. Piranda and J. Bourgeois, “Designing a quasi-spherical module for a huge modular robot to create programmable matter,” *Autonomous Robots*, vol. 42, no. 8, pp. 1619–1633, 2018.

[72] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, “M-TRAN: Self-reconfigurable modular robotic system,” *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 431–441, 2002.

[73] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata, “Distributed Self-Reconfiguration of M-TRAN III Modular Robotic System,” *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 373–386, 2008.

[74] A. Sproewitz, A. Billard, P. Dillenbourg, and A. J. Ijspeert, “Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 4259–4264.

[75] J. Zhao, X. Cui, Y. Zhu, and S. Tang, “A new self-reconfigurable modular robotic system UBot: Multi-mode locomotion and self-reconfiguration,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1020–1025.

[76] K. Gilpin, K. Kotay, and D. Rus, “Miche: Modular Shape Formation by Self-Dissasembly,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 2241–2247.

[77] B. Haghighat, E. Droz, and A. Martinoli, “Lily: A miniature floating robotic platform for programmable stochastic self-assembly,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1941–1948.

[78] J. A. Escalera, M. J. Doyle, F. Mondada, and R. Groß, "Evo-Bots: A Simple, Stochastic Approach to Self-assembling Artificial Organisms," in *Distributed Autonomous Robotic Systems: The 13th International Symposium*, ser. Springer Proceedings in Advanced Robotics, R. Groß *et al.*, Eds., Cham: Springer International Publishing, 2018, pp. 373–385.

[79] T. Fukuda, T. Ueyama, Y. Kawauchi, and F. Arai, "Concept of cellular robotic system (CEBOT) and basic strategies for its realization," *Computers & Electrical Engineering*, vol. 18, no. 1, pp. 11–39, 1992.

[80] M. D. Kutzer, M. S. Moses, C. Y. Brown, M. Armand, D. H. Scheidt, and G. S. Chirikjian, "Design of a new independently-mobile reconfigurable modular robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2758–2764.

[81] K. C. Wolfe, M. S. Moses, M. D. Kutzer, and G. S. Chirikjian, "M3Express: A low-cost independently-mobile reconfigurable modular robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 2704–2710.

[82] G. G. Ryland and H. H. Cheng, "Design of iMobot, an intelligent reconfigurable mobile robot with novel locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 60–65.

[83] C. Liu, Q. Lin, H. Kim, and M. Yim, "SMORES-EP, a modular robot with parallel self-assembly," *Autonomous Robots*, vol. 47, no. 2, pp. 211–228, 2023.

[84] M. J. Doyle, X. Xu, Y. Gu, F. Perez-Diaz, C. Parrott, and R. Groß, "Modular Hydraulic Propulsion: A robot that moves by routing fluid through itself," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5189–5196.

[85] D. Saldaña, B. Gabrich, G. Li, M. Yim, and V. Kumar, "ModQuad: The Flying Modular Structure that Self-Assembles in Midair," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 691–698.

[86] S. Yi, Z. Temel, and K. Sycara, "PuzzleBots: Physical Coupling of Robot Swarms," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 8742–8748.

[87] J. Davey, N. Kwok, and M. Yim, "Emulating self-reconfigurable robots - design of the SMORES system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4464–4469.

[88] R. H. Peck, J. Timmis, and A. M. Tyrrell, "Self-Assembly and Self-Repair during Motion with Modular Robots," *Electronics*, vol. 11, no. 10, p. 1595, 10 2022.

[89] J. Liedke, R. Matthias, L. Winkler, and H. Wörn, "The Collective Self-reconfigurable Modular Organism (CoSMO)," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2013, pp. 1–6.

[90] M. J. Doyle *et al.*, "Modular Fluidic Propulsion Robots," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 532–549, 2021.

[91] R. Oung and R. D'Andrea, "The Distributed Flight Array," *Mechatronics*, vol. 21, no. 6, pp. 908–917, 2011.

[92] R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous Self-Assembly in Swarm-Bots," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1115–1130, 2006.

[93] M. Malley, B. Haghighat, L. Houe, and R. Nagpal, "Eciton robotica: Design and Algorithms for an Adaptive Self-Assembling Soft Robot Collective," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4565–4571.

[94] G. Liang, H. Luo, M. Li, H. Qian, and T. L. Lam, "FreeBOT: A Freeform Modular Self-reconfigurable Robot with Arbitrary Connection Point - Design and Implementation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 6506–6513.

[95] P. Swissler and M. Rubenstein, "FireAnt3D: A 3D self-climbing robot towards non-latticed robotic self-assembly," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 3340–3347.

[96] F. Mondada *et al.*, "Swarm-Bot: A New Distributed Robotic Concept," *Autonomous Robots*, vol. 17, no. 2, pp. 193–221, 2004.

[97] M. Shimizu, T. Mori, and A. Ishiguro, "A Development of a Modular Robot That Enables Adaptive Reconfiguration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 174–179.

[98] M. Malley, M. Rubenstein, and R. Nagpal, "Flippy: A soft, autonomous climber with simple sensing and control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 6533–6540.

[99] Y. Tu, G. Liang, and T. L. Lam, "FreeSN: A Freeform Strut-node Structured Modular Self-reconfigurable Robot - Design and Implementation," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4239–4245.

[100] D. Zhao and T. L. Lam, "SnailBot: A Continuously Dockable Modular Self-reconfigurable Robot Using Rocker-bogie Suspension," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4261–4267.

[101] P. Swissler and M. Rubenstein, "FireAnt: A Modular Robot with Full-Body Continuous Docks," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6812–6817.

[102] J. Liedke and H. Wörn, "CoBoLD — A bonding mechanism for modular self-reconfigurable mobile robots," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011, pp. 2025–2030.

[103] C. H. Belke and J. Paik, "Automatic Couplings With Mechanical Overload Protection for Modular Robots," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 1420–1426, 2019.

[104] A. Spröwitz, R. Moeckel, M. Vespignani, S. Bonardi, and A. J. Ijspeert, "Roombots: A hardware perspective on 3D self-reconfiguration and locomotion with a homogeneous modular robot," *Robotics and Autonomous Systems*, Reconfigurable Modular Robotics, vol. 62, no. 7, pp. 1016–1033, 2014.

[105] W. Saab and P. Ben-Tzvi, "A Genderless Coupling Mechanism With Six-Degrees-of-Freedom Misalignment Capability for Modular Self-Reconfigurable Robots," *Journal of Mechanisms and Robotics*, vol. 8, no. 6, 2016.

[106] A. Sproewitz, M. Asadpour, Y. Bourquin, and A. J. Ijspeert, “An active connection mechanism for modular self-reconfigurable robotic systems based on physical latching,” in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 3508–3513.

[107] S. G. M. Hossain, C. A. Nelson, and P. Dasgupta, “RoGenSiD: A Rotary Plate Genderless Single-Sided Docking Mechanism for Modular Self-Reconfigurable Robots,” presented at the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers Digital Collection, 2014.

[108] B. Salemi, M. Moll, and W.-m. Shen, “SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 3636–3641.

[109] W.-M. Shen, R. Kovac, and M. Rubenstein, “SINGO: A single-end-operative and genderless connector for self-reconfiguration, self-assembly and self-healing,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 4253–4258.

[110] S. S. Sohal, B. Sebastian, and P. Ben-Tzvi, “Autonomous Docking of Hybrid-Wheeled Modular Robots With an Integrated Active Genderless Docking Mechanism,” *Journal of Mechanisms and Robotics*, vol. 14, no. 1, 2021.

[111] E. Klavins, “Programmable Self-Assembly,” *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 43–56, 2007.

[112] B. T. Kirby *et al.*, “A modular robotic system using magnetic force effectors,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 2787–2793.

[113] Y. Peng *et al.*, “A High-Voltage Generator and Multiplexer for Electrostatic Actuation in Programmable Matter,” *IEEE Journal of Solid-State Circuits*, pp. 1–14, 2022.

[114] V. Zykov, E. Mytilinaios, M. Desnoyer, and H. Lipson, “Evolved and Designed Self-Reproducing Modular Robotics,” *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 308–319, 2007.

[115] N. J. Wilson, S. Ceron, L. Horowitz, and K. Petersen, “Scalable and Robust Fabrication, Operation, and Control of Compliant Modular Robots,” *Frontiers in Robotics and AI*, vol. 7, 2020.

[116] T. Tosun, J. Davey, C. Liu, and M. Yim, “Design and characterization of the EP-Face connector,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 45–51.

[117] M. E. Karagozler, J. D. Campbell, G. K. Fedder, S. C. Goldstein, M. P. Weller, and B. W. Yoon, “Electrostatic latching for inter-module adhesion, power transfer, and communication in modular robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 2779–2786.

[118] M. E. Karagozler, S. C. Goldstein, and J. R. Reid, “Stress-driven MEMS assembly + electrostatic forces = 1mm diameter robot,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 2763–2769.

[119] J. Neubert, A. Rost, and H. Lipson, “Self-Soldering Connectors for Modular Robots,” *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1344–1357, 2014.

[120] L. S. Penrose and R. Penrose, “A Self-reproducing Analogue,” *Nature*, vol. 179, no. 4571, pp. 1183–1183, 4571 1957.

[121] M. Jilek, M. Somr, M. Kulich, J. Zeman, and L. Preucil, “Towards a passive self-assembling macroscale multi-robot system,” *IEEE Robotics and Automation Letters*, pp. 1–1, 2021.

[122] K. Stoy, W.-M. Shen, and P. Will, “Using role-based control to produce locomotion in chain-type self-reconfigurable robots,” *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 410–417, 2002.

[123] W.-M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh, “Multimode locomotion via SuperBot reconfigurable robots,” *Autonomous Robots*, vol. 20, no. 2, pp. 165–177, 2006.

[124] G. Jing, T. Tosun, M. Yim, and H. Kress-Gazit, “An End-To-End System for Accomplishing Tasks with Modular Robots,” presented at the Robotics: Science and Systems XII, vol. 12, 2016.

[125] D. Brandt, D. J. Christensen, and H. H. Lund, “ATRON Robots: Versatility from Self-Reconfigurable Modules,” in *2007 International Conference on Mechatronics and Automation*, 2007, pp. 26–32.

[126] J. W. Romanishin, K. Gilpin, and D. Rus, “M-blocks: Momentum-driven, magnetic modular robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 4288–4295.

[127] D. Duff, M. Yim, and K. Roufas, “Evolution of polybot: A modular reconfigurable robot,” in *Proc. of the Harmonic Drive Intl. Symposium, Nagano, Japan*, 2001.

[128] J. White, M. A. Post, and A. Tyrrell, “A Novel Connection Mechanism for Dynamically Reconfigurable Modular Robots,” in *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, IEEE, 2022.

[129] C.-A. Chen, A. Kamimura, L. Barrios, and W.-M. Shen, “Dynamic Power Sharing for Self-Reconfigurable Modular Robots,” in *Towards Autonomous Systems (TAROS)*, Springer, 2014, pp. 3–14.

[130] K. Holdcroft, C. H. Belke, S. Bennani, and J. Paik, “3PAC: A plug-and-play system for distributed power sharing and communication in modular robots,” *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 2, pp. 858–867, 2022.

[131] G. Liang, Y. Tu, L. Zong, J. Chen, and T. L. Lam, “Energy Sharing Mechanism for a Freeform Robotic System - FreeBOT,” in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4232–4238.

[132] K. Holdcroft, A. Bolotnikova, C. Belke, and J. Paik, “Modular robot networking: A novel schema and its performance assessment,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 12 698–12 705.

[133] P. Thalamy, B. Piranda, and J. Bourgeois, "A survey of autonomous self-reconfiguration methods for robot-based programmable matter," *Robotics and Autonomous Systems*, vol. 120, p. 103 242, 2019.

[134] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[135] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Automatic locomotion design and experiments for a Modular robotic system," *IEEE/ASME Transactions on Mechatronics*, vol. 10, no. 3, pp. 314–325, 2005.

[136] K. Tomita, S. Murata, H. Kurokawa, E. Yoshida, and S. Kokaji, "Self-assembly and self-repair method for a distributed mechanical system," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 6, pp. 1035–1045, 1999.

[137] M. Yim, Y. Zhang, J. Lamping, and E. Mao, "Distributed Control for 3D Metamorphosis," *Autonomous Robots*, vol. 10, no. 1, pp. 41–56, 2001.

[138] D. Brandt, "Comparison of A* and RRT-Connect Motion Planning Techniques for Self-Reconfiguration Planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 892–897.

[139] A. Sproewitz *et al.*, "Roombots - Towards decentralized reconfiguration with self-reconfiguring modular robotic metamodules," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 1126–1132.

[140] R. Fitch, Z. Butler, and D. Rus, "Reconfiguration planning for heterogeneous self-reconfiguring robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2003, pp. 2460–2467.

[141] E. Hourany, C. Stephan, A. Makhoul, B. Piranda, B. Habib, and J. Bourgeois, "Self-Reconfiguration of Modular Robots Using Virtual Forces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6948–6953.

[142] Y. Zhu, D. Bie, X. Wang, Y. Zhang, H. Jin, and J. Zhao, "A distributed and parallel control mechanism for self-reconfiguration of modular robots using L-systems and cellular automata," *Journal of Parallel and Distributed Computing*, vol. 102, pp. 80–90, 2017.

[143] H.-a. Yang, S. Cao, L. Bai, Z. Zhang, and J. Kong, "A distributed and parallel self-assembly approach for swarm robotics," *Robotics and Autonomous Systems*, vol. 118, pp. 80–92, 2019.

[144] H. Kawano, "Complete reconfiguration algorithm for sliding cube-shaped modular robots with only sliding motion primitive," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 3276–3283.

[145] H. Kawano, "Distributed Linear Heterogeneous Reconfiguration of Cubic Modular Robots via Simultaneous Tunneling and Permutation," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 62–77, 2020.

[146] Z. Ye, M. Yu, and Y.-J. Liu, "NP-completeness of optimal planning problem for modular robots," *Autonomous Robots*, vol. 43, no. 8, pp. 2261–2270, 2019.

[147] F. Hou and W.-M. Shen, “On the complexity of optimal reconfiguration planning for modular reconfigurable robots,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 2791–2796.

[148] C. Sung, J. Bern, J. Romanishin, and D. Rus, “Reconfiguration planning for pivoting cube modular robots,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1933–1940.

[149] D. Feshbach and C. Sung, “Reconfiguring Non-Convex Holes in Pivoting Modular Cube Robots,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6701–6708, 2021.

[150] J. W. Romanishin, J. Mamish, and D. Rus, “Decentralized Control for 3D M-Blocks for Path Following, Line Formation, and Light Gradient Aggregation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 4862–4868.

[151] C. Liu, M. Whitzer, and M. Yim, “A Distributed Reconfiguration Planning Algorithm for Modular Robots,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4231–4238, 2019.

[152] L. Zhang, Z.-H. Fu, H. Liu, Q. Liu, X. Ji, and H. Qian, “An Efficient Parallel Self-assembly Planning Algorithm for Modular Robots in Environments with Obstacles,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 10 038–10 044.

[153] P. Thalamy, B. Piranda, and J. Bourgeois, “Engineering efficient and massively parallel 3D self-reconfiguration using sandboxing, scaffolding and coating,” *Robotics and Autonomous Systems*, vol. 146, p. 103 875, 2021.

[154] P. Thalamy, B. Piranda, F. Lassabe, and J. Bourgeois, “Deterministic scaffold assembly by self-reconfiguring micro-robotic swarms,” *Swarm and Evolutionary Computation*, vol. 58, p. 100 722, 2020.

[155] P. Thalamy, B. Piranda, and J. Bourgeois, “3D Coating Self-Assembly for Modular Robotic Scaffolds,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 11 688–11 695.

[156] J. Bassil, B. Piranda, A. Makhoul, and J. Bourgeois, “RePoSt: Distributed Self-Reconfiguration Algorithm for Modular Robots Based on Porous Structure,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 12 651–12 658.

[157] I. Parada, V. Sacristán, and R. I. Silveira, “A new meta-module design for efficient reconfiguration of modular robots,” *Autonomous Robots*, vol. 45, no. 4, pp. 457–472, 2021.

[158] E. Klavins, R. Ghrist, and D. Lipsky, “Graph grammars for self assembling robotic systems,” in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 5, 2004, pp. 5293–5300.

[159] B. Haghighat and A. Martinoli, “Automatic synthesis of rulesets for programmable stochastic self-assembly of rotationally symmetric robotic modules,” *Swarm Intelligence*, vol. 11, no. 3, pp. 243–270, 2017.

[160] B. Haghighat, R. Thandiackal, M. Mordig, and A. Martinoli, "Probabilistic modeling of programmable stochastic self-assembly of robotic modules," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4656–5663.

[161] M. Gauci, R. Nagpal, and M. Rubenstein, "Programmable Self-disassembly for Shape Formation in Large-Scale Robot Collectives," in *Distributed Autonomous Robotic Systems: 13th International Symposium*, Cham: Springer International Publishing, 2018, pp. 573–586.

[162] M. D. Hall, A. Özdemir, and R. Groß, "Self-Reconfiguration in Two-Dimensions via Active Subtraction with Modular Robots," in *Robotics: Science and Systems*, 2020.

[163] H. Bojinov, A. Casal, and T. Hogg, "Multiagent control of self-reconfigurable robots," *Artificial Intelligence*, International Conference on MultiAgent Systems 2000, vol. 142, no. 2, pp. 99–120, 2002.

[164] D. Christensen, "Evolution of shape-changing and self-repairing control for the ATRON self-reconfigurable robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 2539–2545.

[165] R. Fitch and Z. Butler, "Million Module March: Scalable Locomotion for Large Self-Reconfiguring Robots," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 331–343, 2008.

[166] Y. Zhu, D. Bie, S. Iqbal, X. Wang, Y. Gao, and J. Zhao, "A Simplified Approach to Realize Cellular Automata for UBot Modular Self-Reconfigurable Robots," *Journal of Intelligent & Robotic Systems*, vol. 79, no. 1, pp. 37–54, 2015.

[167] H. Luo, M. Li, G. Liang, H. Qian, and T. L. Lam, "An Obstacle-crossing Strategy Based on the Fast Self-reconfiguration for Modular Sphere Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 3296–3303.

[168] H. Luo and T. L. Lam, "Adaptive Flow Planning of Modular Spherical Robot Considering Static Gravity Stability," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4228–4235, 2022.

[169] R. O'Grady, R. Groß, A. L. Christensen, and M. Dorigo, "Self-assembly strategies in a group of autonomous mobile robots," *Autonomous Robots*, vol. 28, no. 4, pp. 439–455, 2010.

[170] E. Gonzalez, L. Houel, R. Nagpal, and M. Malley, "Influencing Emergent Self-Assembled Structures in Robotic Collectives Through Traffic Control," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 2022, pp. 1601–1603.

[171] J. Assaker, A. Makhoul, J. Bourgeois, and J. Demerjian, "A Unique Identifier Assignment Method for Distributed Modular Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 3304–3311.

[172] J. Assaker, A. Makhoul, J. Bourgeois, B. Piranda, and J. Demerjian, "A Dynamic ID Assignment Approach for Modular Robots," in *Advanced Information Networking and Applications*, L. Barolli, F. Hussain, and T. Enokido, Eds., ser. Lecture Notes in Networks and Systems, Cham: Springer International Publishing, 2022, pp. 91–104.

[173] J. Baca, B. Woosley, P. Dasgupta, and C. A. Nelson, "Configuration discovery of modular self-reconfigurable robots: Real-time, distributed, IR+XBee communication method," *Robotics and Autonomous Systems*, vol. 91, pp. 284–298, 2017.

[174] C. Liu and M. Yim, "Configuration Recognition with Distributed Information for Modular Robots," in *Robotics Research*, N. M. Amato, G. Hager, S. Thomas, and M. Torres-Torriti, Eds., ser. Springer Proceedings in Advanced Robotics, Cham: Springer International Publishing, 2020, pp. 967–983.

[175] Y. Tu, G. Liang, and T. L. Lam, "Graph Convolutional Network based Configuration Detection for Freeform Modular Robot Using Magnetic Sensor Array," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4252–4258.

[176] J. Bassil, P. Tannoury, B. Piranda, A. Makhoul, and J. Bourgeois, "Fault-Tolerance Mechanism for Self-Reconfiguration of Modular Robots," in *International Wireless Communications and Mobile Computing (IWCMC)*, 2022, pp. 360–365.

[177] B. Piranda, P. Chodkiewicz, P. Hołobut, S. P. A. Bordas, J. Bourgeois, and J. Lengiewicz, "Distributed Prediction of Unsafe Reconfiguration Scenarios of Modular Robotic Programmable Matter," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2226–2233, 2021.

[178] Y. Suzuki, N. Inou, H. Kimura, and M. Koseki, "Reconfigurable group robots adaptively transforming a mechanical structure - Crawl motion and adaptive transformation with new algorithms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 2200–2205.

[179] Y. Suzuki, N. Inou, H. Kimura, and M. Koseki, "Reconfigurable group robots adaptively transforming a mechanical structure - numerical expression of criteria for structural transformation and automatic motion planning method -," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 2361–2367.

[180] N. Melenbrink, P. Michalatos, P. Kassabian, and J. Werfel, "Using local force measurements to guide construction by distributed climbing robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 4333–4340.

[181] N. Melenbrink and J. Werfel, "Local force cues for strength and stability in a distributed robotic construction system," *Swarm Intelligence*, vol. 12, no. 2, pp. 129–153, 2017.

[182] Engineering Toolbox. "Cantilever Beams - Moments and Deflections." (2013), [Online]. Available: `https://www.engineeringtoolbox.com/cantilever-beams-d_1848.html` (visited on Dec. 8, 2022).

[183] E. Bray and R. Groß, "Distributed Self-Assembly of Cantilevers by Force-Aware Robots," in *International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2021, pp. 110–118.

[184] *Webots: Open source robot simulator*, Cyberbotics Ltd. [Online]. Available: `https://cyberbotics.com/#cyberbotics` (visited on Feb. 1, 2023).

[185] E. Rohmer, S. P. N. Singh, and M. Freese, "CoppeliaSim (formerly V-REP): A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326.

[186] R. Vink, *anaStruct*, version 1.1.2, 2020. [Online]. Available: `https://github.com/ritchie46/anaStruct`.

[187] D. Guichard, *An Introduction to Combinatorics and Graph Theory*. Whitman College-Creative Commons, 2017.

[188] S. H. Crandall, N. C. Dahl, and T. J. Lardner, *An Introduction to the Mechanics of Solids*, 2nd edition with SI units. McGraw-Hill, 1978.

[189] Engineering Toolbox. "Beams - Fixed at Both Ends - Continuous and Point Loads." (2004), [Online]. Available: `https://www.engineeringtoolbox.com/beams-fixed-both-ends-support-loads-deflection-d_809.html` (visited on Nov. 15, 2022).

[190] E. Bray and R. Groß, "Distributed Optimisation and Deconstruction of Bridges by Self-Assembling Robots," in *Robotics: Science and Systems*, 2022.

[191] *Arduino Core for mbed enabled devices*, Arduino, 2020. [Online]. Available: `https://github.com/arduino/ArduinoCore-mbed` (visited on Jan. 19, 2023).

[192] *QThread - Qt for Python*, PyQT. [Online]. Available: `https://doc.qt.io/qtforpython/PySide6/QtCore/QThread.html` (visited on Jan. 19, 2023).

[193] S. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," *Report x-io and University of Bristol (UK)*, vol. 25, pp. 113–118, 2010.

[194] J. Seitz, "Autonomous design of modular robot extensions using an evolutionary algorithm," BSc Thesis, ETH Zürich, Zürich, Switzerland, 2013.

[195] *Abaqus Unified FEA*, version 2017, Dassault Systèmes Simulia Corporation.

[196] W. D. Pilkey, *Peterson's Stress Concentration Factors*, 2nd ed. John Wiley & Sons, 1997.

Appendix A

Simulator Design

This appendix describes how the simulator calculates forces within links between modules by approximating the structure as a truss. It is inspired by the method used by Brodbeck and Iida [16], which is explained in more detail in [194].

Two additional methods of calculating the forces within a 2D grid of connected agents were also considered. A first simulator used elastic beam theory to calculate the forces by considering the structure as an idealised beam [188]. However, this method was not able to accurately capture how the forces varied due to the discrete nature of the structure. A second simulator used finite element analysis implemented with Abaqus [195] to calculate these forces. It was found that this method could calculate a more appropriate force distribution than using elastic beam theory, but took significantly longer and had issues where stress concentrations occurred around the sharp corners formed by the discrete agents [196]. The truss based approach described here is able to accurately calculate the force distribution for these discrete structures in a reasonable amount of time, typically taking less than 5 s, even for structures consisting of over 100 agents.

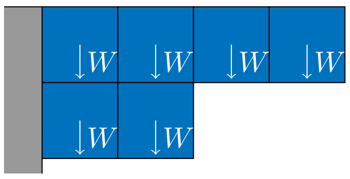
A.1 Creating a Truss

The structure is represented as a two-dimensional truss. Each agent is modelled as a square of truss members with a cross-bracings on both diagonals, giving each agent six truss members in total ($N_{agent} = 6$). The weight of the agent W is given by:

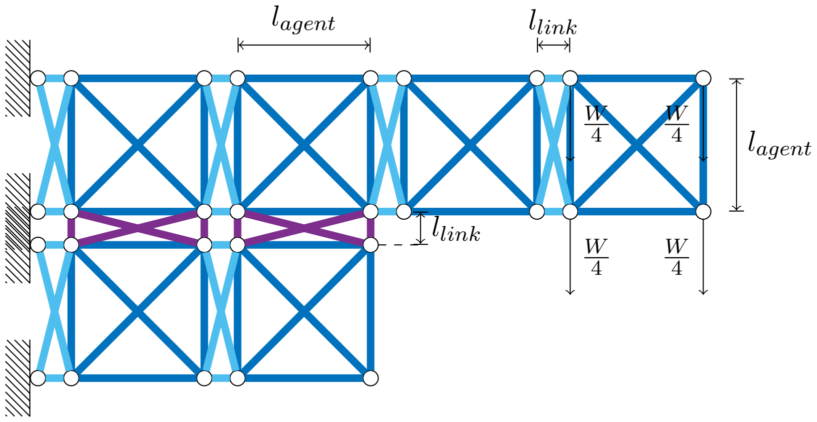
$$W = mg \tag{A.1}$$

where m is the mass of each agent, and g is the acceleration due to gravity, taken to be $9.81\,\mathrm{m\,s^{-2}}$. This would normally act vertically down through the centre of mass of the agent, but here it is split evenly between the four corner nodes. Links between agents are also modelled as cross-braced trusses, but the faces attached to the agents already have truss members due to the agents themselves, so each link only contains four truss members ($N_{link} = 4$). Links are modelled as light, so have no weight. An example of how a cantilever shape of six agents is modelled as a truss is shown in Figure A.1.

This modelling is implemented in Python using the anaStruct package [186], which allows trusses to be easily created and the internal forces examined. Agents are taken as cubes of side length $l_{agent} = 0.09\,\mathrm{m}$ made of solid aluminium, which has density $\rho =$



(a)



(b)

Figure A.1. Conversion of a cantilever of six agents into a truss. **(a)** The cantilever shape, where each agent has weight W: agents are shown blue, and the fixed support in grey. **(b)** The truss approximation, showing agents as blue members, links along rows in cyan, and links down columns in purple: weight is distributed by a force of magnitude $\frac{W}{4}$ acting downwards from each corner of the agent trusses (only shown once).

2700 kg m^{-3} and Young's modulus $E = 69$ GPa. Therefore W is therefore calculated as:

$$W = \rho l_{agent}^3 g \tag{A.2}$$

This gives $W = 19.3$ N.

Links between agents are modelled as having a length $l_{link} = 0.01$ m. They have the same Young's Modulus as agents, but are assumed to be light in comparison to them hence have no mass. They also have the same cross-sectional area on the contact face. The angle between the diagonal members in the links and the ones between agents is denoted θ.

The length of each truss member is dependent on the node locations, but the cross-sectional area must be specified. This is approximated so that the sum of the areas of the members in each agent A_{agent} or link A_{link} is equal to the cross-sectional area of the contact face:

$$A_{agent} = \frac{l_{agent}^2}{N_{agent}} \tag{A.3}$$

$$A_{link} = \frac{l_{agent}^2}{N_{link}} \tag{A.4}$$

A.2 Calculating Forces and Moments

The model is now defined, and the internal forces within each member can be calculated. This is done with the `anastruct.solve()` function, which calculates the force in each truss member. Cuts are made through the middle of each link as shown in Figure A.2 to reveal these internal forces $f_i \; \forall \; 1 \leq i \leq 4$. Since pin-jointed truss members can

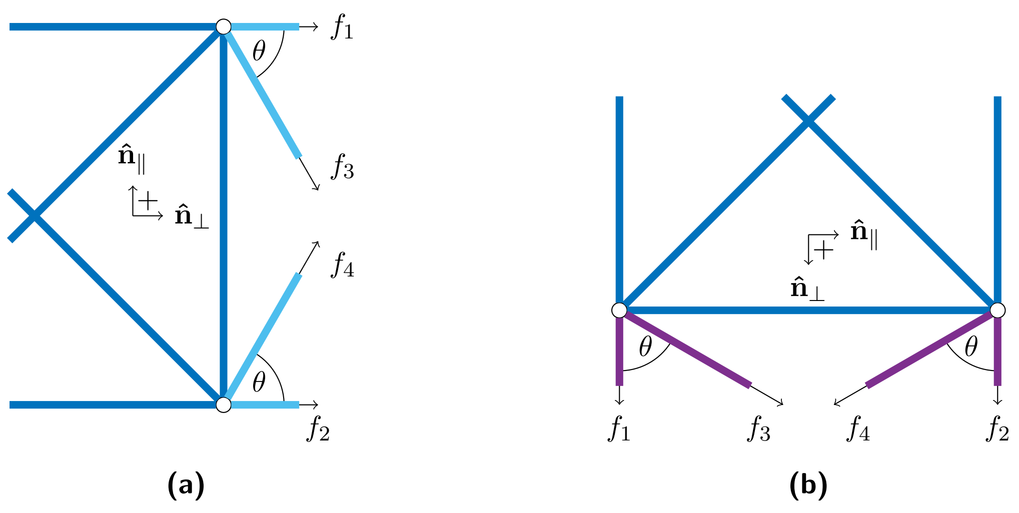


Figure A.2. Cuts made to reveal internal forces in links along **(a)** rows and **(b)** columns. Cyan and purple truss members model the links, and blue truss members model the agents. The directions of the unit vectors $\hat{\mathbf{n}}_{\perp}$ and $\hat{\mathbf{n}}_{||}$ are also shown.

only carry axial force, the direction of the force can be obtained by trigonometry to give the vectors $\boldsymbol{F}_i$. In links along rows, these are given by:

$$\boldsymbol{F}_1 = \begin{bmatrix} f_1 & 0 \end{bmatrix}^T \qquad \boldsymbol{F}_3 = \begin{bmatrix} f_3 \cos\theta & -f_3 \sin\theta \end{bmatrix}^T \tag{A.5}$$

$$\boldsymbol{F}_2 = \begin{bmatrix} f_2 & 0 \end{bmatrix}^T \qquad \boldsymbol{F}_4 = \begin{bmatrix} f_4 \cos\theta & f_4 \sin\theta \end{bmatrix}^T \tag{A.6}$$

And in links between columns by:

$$\boldsymbol{F}_1 = \begin{bmatrix} 0 & -f_1 \end{bmatrix}^T \qquad \boldsymbol{F}_3 = \begin{bmatrix} f_3 \sin\theta & -f_3 \cos\theta \end{bmatrix}^T \tag{A.7}$$

$$\boldsymbol{F}_2 = \begin{bmatrix} 0 & -f_2 \end{bmatrix}^T \qquad \boldsymbol{F}_4 = \begin{bmatrix} -f_4 \sin\theta & -f_4 \cos\theta \end{bmatrix}^T \tag{A.8}$$

The total force vector is then calculated as:

$$\boldsymbol{F}_{tot} = \sum_{i=1}^{4} \boldsymbol{F}_i \tag{A.9}$$

This vector can be resolved into a longitudinal component normal to the plane $F_{\perp}$ and a shear one parallel to it $F_{||}$ by multiplying by the corresponding unit vectors $\hat{\mathbf{n}}_{\perp}$ and $\hat{\mathbf{n}}_{||}$ respectively:

$$F_{\perp} = \boldsymbol{F}_{tot} \cdot \hat{\boldsymbol{n}}_{\perp} \tag{A.10}$$

$$F_{||} = \boldsymbol{F}_{tot} \cdot \hat{\boldsymbol{n}}_{||} \tag{A.11}$$

The force in each truss member also causes a moment in the link. The moment is taken about the centre of the link by calculating the cross product of the perpendicular vector from the centre to the line of action of $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ ($\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ respectively, giving the moments $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ respectively. $\boldsymbol{F}_3$ and $\boldsymbol{F}_4$ can be neglected as they act through the centre of the link and therefore cause no moment. These are summed to get the total moment, $\boldsymbol{M}_{tot}$:

$$\begin{aligned}\boldsymbol{M}_{tot} &= \boldsymbol{M}_1 + \boldsymbol{M}_2 \\ \boldsymbol{M}_{tot} &= (\boldsymbol{r}_1 \times \boldsymbol{F}_1) + (\boldsymbol{r}_2 \times \boldsymbol{F}_2) \qquad \text{(A.12)}\end{aligned}$$

The work of Seitz converts these forces and moments to an equivalent Von Mises stress [194]. This step is not taken here. Instead the simulator uses the axial force $F_\perp$ defined as positive when force acts to pull the links apart, and the magnitude of the moment $|\boldsymbol{M}_{tot}|$. These values are given the symbols F and M respectively outside this appendix.

A.3 Validation

The results are validated by considering a simple cantilever consisting of ten agents in a straight line (Figure A.3a). In this example, static equilibrium can be used to calculate the values of shear force $F_\parallel$ and moment $\boldsymbol{M}_{tot} = M$ at different points along the beam; the axial force $F_\perp$ will always be 0 in this scenario. Figure A.3b shows a cut made in the beam n agents from the tip. Each agent has a side length $l = l_{agent} + l_{link}$. From static equilibrium, it can be seen that:

$$F_\parallel = nW \qquad \text{(A.13)}$$

$$M = nW \cdot \frac{nl}{2} \qquad \text{(A.14)}$$

For this system, $W = 19.3\,\mathrm{N}$ and $l = 0.1\,\mathrm{m}$.

Comparisons of $F_\parallel$ and M as calculated by static equilibrium and the truss approximation can be made for this simple structure, shown in Figures A.3c & A.3d respectively. The two methods produce very similar data, with minor differences occurring due to rounding errors in the calculations. This analysis is only possible for a cantilever of continuous height, but it still demonstrates the accuracy of the truss approximation method used by the simulations in this thesis.

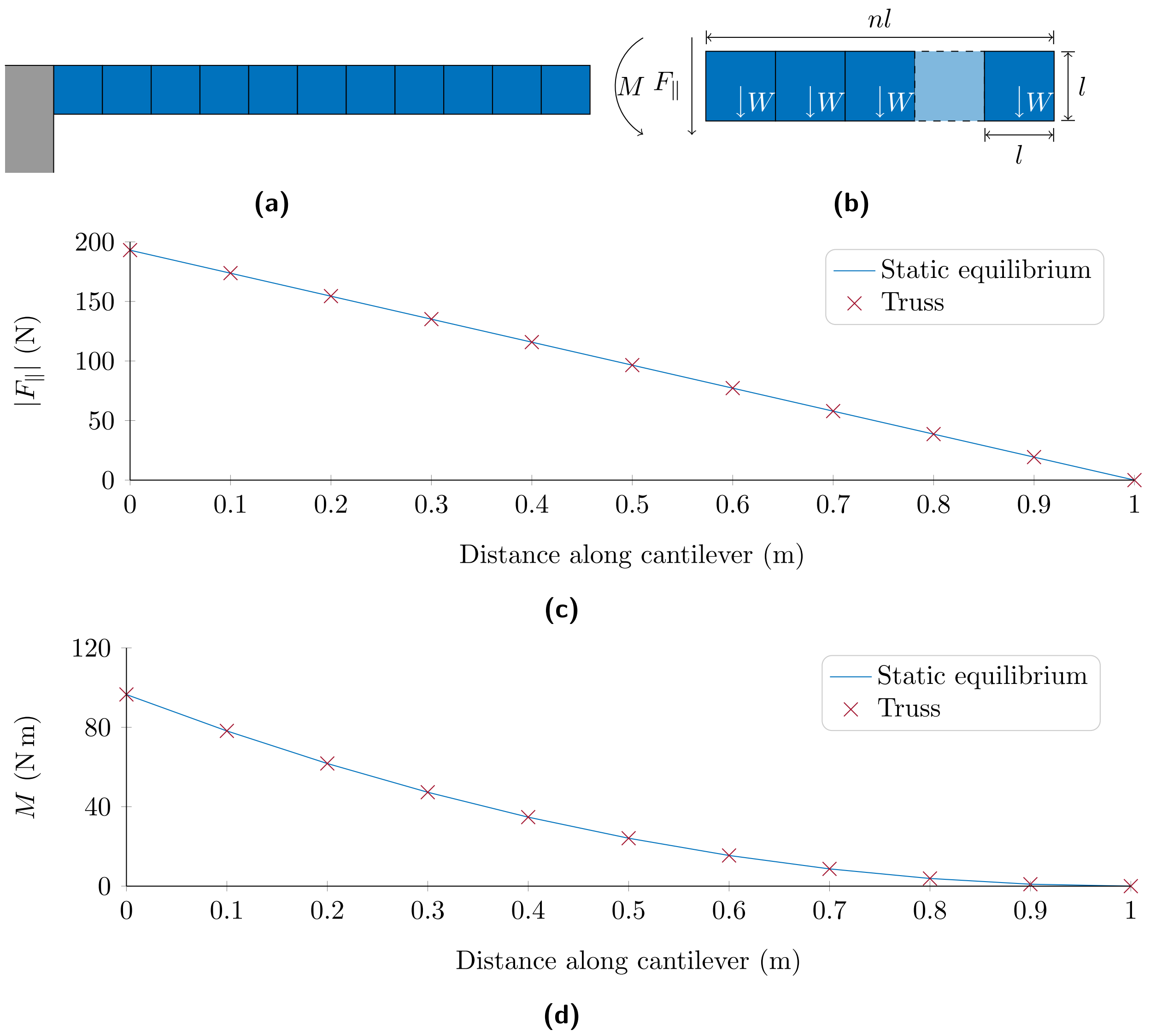


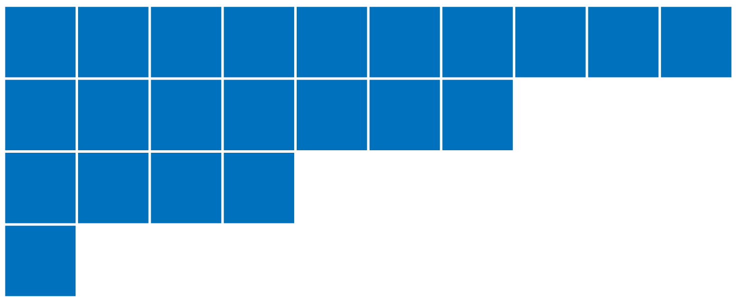
Figure A.3. A comparison between theoretically calculated values of shear force S and moment M in a simple structure. **(a)** The structure, with agents shown in blue and the fixed support in grey. **(b)** A cut through this structure n agents from the tip to reveal the moment M and shear force $F_{\|}$ at this point. **(c & d)** The distribution of $|F_{\|}|$ and M respectively within the structure as calculated by static equilibrium and the truss approximation method.

Appendix B

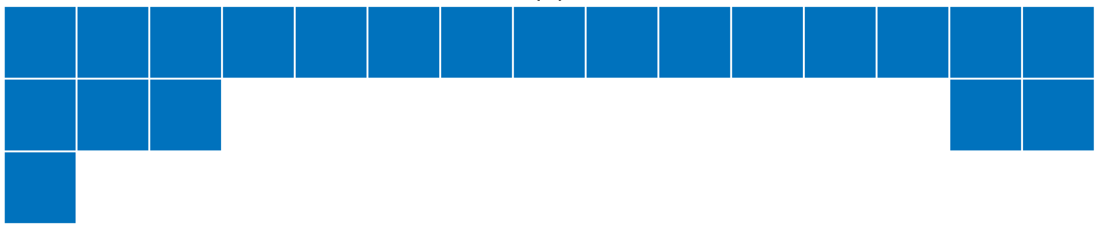
Optimal Structures

This appendix contains the optimal configurations of cantilevers and bridges, calculated for weak, medium, and strong links. Configurations are given for the minimum number of agents that can achieve a cantilever or bridge of a given length without any critical links.

Each configuration is represented as a string listing the numbers of agents in each row in descending order from row 1. The rows are separated by slashes. In cantilevers, all agents are connected to a single support so the number of agents in each row is sufficient information to represent the whole structure. However, bridges are connected to both supports, so each row is split with an underscore to denote the number of agents connected to the left and right supports respectively; each top row is denoted as `X_0` where `X` is the number of agents in this row, which is connected to both supports. Examples of these representations are given in Figure B.1.



(a)



(b)

Figure B.1. Examples of structures and their string representation. **(a)** A cantilever with string representation `10/7/4/1`, and **(b)** a bridge with string representation `15_0/3_2/1_0`

B.1 Optimal Cantilevers

Table B.1. Optimal cantilevers with weak links.

Length (bodylengths)	Minimum number of agents	Optimal configuration(s)
1	1	1
2	2	2
3	3	3
4	5	4/1
5	7	5/2
6	9	6/3
7	11	7/4
8	15	8/5/2
9	18	9/6/3
10	22	10/7/4/1
11	27	11/8/5/3, 11/8/5/2/1
12	33	12/9/6/4/2
13	41	13/10/7/5/4/2, 13/10/7/5/3/2/1
14	51	14/11/8/6/5/4/3, 14/11/8/6/5/4/2/1, 14/11/8/5/5/4/3/1, 14/11/8/5/4/4/3/2, 14/11/8/5/4/3/3/2/1
15	62	15/12/9/7/6/5/4/3/1, 15/12/9/6/5/5/4/3/2/1, 15/12/9/6/6/5/4/3/2
16	76	16/13/10/7/7/6/5/4/4/3/1, 16/13/10/8/7/6/5/4/3/3/1, 16/13/10/8/7/6/5/4/3/2/2, 16/13/10/7/7/6/5/4/3/3/2, 16/13/10/7/7/6/5/4/3/3/1/1, 16/13/10/8/7/6/5/4/3/2/1/1, 16/13/10/7/7/6/5/4/3/2/2/1, 16/13/10/7/7/6/5/5/4/2/1, 16/13/10/7/6/6/5/5/4/3/1, 16/13/10/7/6/6/5/4/3/3/2/1, 16/13/10/7/6/6/5/4/4/3/2
17	93	17/14/11/8/8/7/7/6/5/4/3/2/1

Table B.2. Optimal cantilevers with links of medium strength.

Length (bodylengths)	Minimum number of agents	Optimal configuration(s)
1	1	`1`
2	2	`2`
3	3	`3`
4	4	`4`
5	5	`5`
6	6	`6`
7	8	`7/1`
8	10	`8/2`
9	12	`9/3`
10	14	`10/4`
11	16	`11/5`
12	18	`12/6`
13	22	`13/7/2`
14	25	`14/8/3`
15	29	`15/9/4/1`
16	33	`16/10/5/2`
17	38	`17/11/6/3/1`
18	43	`18/12/7/4/2`
19	50	`19/13/8/5/4/1`, `19/13/8/5/3/2`, `19/13/8/4/4/2`, `19/13/8/4/3/3`, `19/13/8/4/3/2/1`
20	56	`20/14/9/6/4/3`, `20/14/9/6/4/2/1`, `20/14/9/5/4/3/1`
21	63	`21/15/10/6/5/4/2`, `21/15/10/6/5/3/2/1`
22	71	`22/16/11/7/6/5/4`, `22/16/11/7/6/5/3/1`, `22/16/11/7/5/5/3/2`, `22/16/11/7/5/4/3/2/1`, `22/16/11/7/6/4/3/2`
23	79	`23/17/12/8/6/5/4/3/1`
24	89	`24/18/13/9/8/6/5/4/2`, `24/18/13/9/7/6/5/4/3`, `24/18/13/9/7/6/5/4/2/1`, `24/18/13/9/7/7/5/4/2`
25	99	`25/19/14/10/8/7/6/5/3/2`

Table B.3. Optimal cantilevers with strong links.

Length (bodylengths)	Minimum number of agents	Optimal configuration(s)
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	11	10/1
11	13	11/2
12	15	12/3
13	17	13/4
14	19	14/5
15	21	15/6
16	23	16/7
17	27	17/8/2
18	30	18/9/3, 18/9/2/1
19	33	19/10/3/1
20	37	20/11/4/2
21	41	21/12/5/3
22	46	22/13/6/4/1, 22/13/6/3/2
23	51	23/14/7/5/2, 23/14/7/4/2/1, 23/14/7/4/3
24	56	24/15/8/5/3/1, 24/15/8/5/4
25	62	25/16/9/6/4/2
26	68	26/17/10/7/4/3/1, 26/17/10/7/5/3
27	75	27/18/11/8/6/4/1, 27/18/11/8/6/3/2, 27/18/11/8/5/4/2
28	82	28/19/12/9/6/4/3/1, 28/19/12/9/6/5/3
29	90	29/20/13/10/7/6/3/2, 29/20/13/10/7/5/4/2, 29/20/13/10/7/6/4/1, 29/20/13/10/7/5/3/2/1
30	98	30/21/14/11/8/6/5/3, 30/21/14/11/8/6/4/3/1

B.2 Optimal Bridges

Table B.4. Optimal bridges with weak links.

Length (bodylengths)	Minimum number of agents	Optimal configuration(s)
1	1	1_0
2	2	2_0
3	3	3_0
4	4	4_0
5	5	5_0
6	6	6_0
7	7	7_0
8	8	8_0
9	9	9_0
10	12	10_0/2_0, 10_0/1_1, 10_0/0_2
11	13	11_0/1_1
12	14	12_0/1_1
13	17	13_0/2_2
14	18	14_0/2_2
15	21	15_0/3_2/1_0, 15_0/3_3, 15_0/2_3/0_1
16	22	16_0/3_3
17	25	17_0/4_3/1_0, 17_0/3_4/0_1
18	28	18_0/4_4/1_1
19	32	19_0/5_4/3_1, 19_0/5_4/2_2, 19_0/4_5/1_3, 19_0/4_5/2_2
20	36	20_0/6_4/4_1/1_0, 20_0/6_5/3_2, 20_0/5_5/4_2, 20_0/5_5/3_2/1_0, 20_0/5_5/3_2/0_1, 20_0/5_5/3_3, 20_0/5_5/2_3/1_0, 20_0/4_6/1_4/0_1, 20_0/5_6/2_3, 20_0/5_5/2_4, 20_0/5_5/2_3/0_1, 20_0/5_5/2_3/1_0, 20_0/5_5/3_2/0_1
21	39	21_0/6_5/4_3, 21_0/6_6/3_3, 21_0/5_6/3_4

Table B.5. Optimal bridges with links of medium strength.

Length (bodylengths)	Minimum number of agents	Optimal configuration(s)
1	1	1_0
2	2	2_0
3	3	3_0
4	4	4_0
5	5	5_0
6	6	6_0
7	7	7_0
8	8	8_0
9	9	9_0
10	10	10_0
11	11	11_0
12	12	12_0
13	13	13_0
14	14	14_0
15	15	15_0
16	16	16_0
17	19	17_0/2_0, 17_0/1_1, 17_0/0_2
18	20	18_0/1_1
19	22	19_0/2_1, 19_0/1_2
20	24	20_0/3_1, 20_0/2_2, 20_0/1_3
21	25	21_0/2_2
22	28	22_0/4_2, 22_0/3_2/1_0, 22_0/3_3, 22_0/2_4, 22_0/2_3/0_1
23	29	23_0/3_3
24	31	24_0/4_3, 24_0/3_4
25	33	25_0/4_4
26	35	26_0/5_4, 26_0/4_5
27	37	27_0/5_5
28	38	28_0/5_5

Table B.6. Optimal bridges with strong links.

Length (bodylengths)	Minimum number of agents	Optimal configuration(s)
1	1	1_0
2	2	2_0
3	3	3_0
4	4	4_0
5	5	5_0
6	6	6_0
7	7	7_0
8	8	8_0
9	9	9_0
10	10	10_0
11	11	11_0
12	12	12_0
13	13	13_0
14	14	14_0
15	15	15_0
16	16	16_0
17	17	17_0
18	18	18_0
19	19	19_0
20	20	20_0
21	21	21_0
22	22	22_0
23	23	23_0
24	26	24_0/2_0, 24_0/1_1, 24_0/0_2
25	27	25_0/1_1
26	29	26_0/2_1, 26_0/1_2
27	31	27_0/3_1, 27_0/2_2, 27_0/1_3
28	32	28_0/2_2
29	35	29_0/4_2, 29_0/3_3, 29_0/2_4
30	36	30_0/3_3
31	38	31_0/4_3, 31_0/3_4
32	40	32_0/4_4

Appendix C

Superposition

In order to improve the performance of the bridge deconstruction algorithm (Chapter 5), the agents should be able to determine what the forces within the links of the structure would be if it was only supported at one end. However, they receive measurements of forces within the structure supported at both ends. Based on the information the agents are able to gather, the concept of *superposition* is used to predict what these values would be. Assuming the structure is made of a homogeneous linear elastic material and deformations are small, these deformations and the force distribution within a structure loaded by multiple forces simultaneously will be equal to the sum of the deformations and force distributions were the structure to be loaded by each of these forces individually [188].

The superimposed cases are shown in Figure C.1. The agents measure force information in the structure labelled *measured*, which is supported at both ends and loaded by an arbitrary distributed load. The *desired* case is for the same structure under the same arbitrary distributed load, but only supported one the right side. The force distribution in the desired case can be found by subtracting the force distributions under the horizontal and vertical reaction forces, R_h and R_v respectively, and the reaction moment R_M from the measured case. These cases are referred to as cases ① – ③ respectively.

If the structure is assumed to be comprised of a homogeneous linear elastic material, elastic beam theory can be used to calculate the force distribution in cases ① – ③ [188]. The structure is modelled as shown in Figure C.2: the height h is assumed to be a continuous function of the distance from the right support x, and the breadth of the structure b is constant. At a distance x from the support, the second moment of area I is a function of x:

$$I = \frac{bh(x)^3}{12} \tag{C.1}$$

Each of the cases is analysed below. In each case, the analysis begins with calculating the distribution of the longitudinal stress σ within the structure based on this loading as a function of x and the distance from the neutral axis y. The structure with continuously-varying height approximates a structure made of discrete square agents of side length l, so the links have a cross-sectional area A_{link} given by:

$$A_{link} = bl \tag{C.2}$$

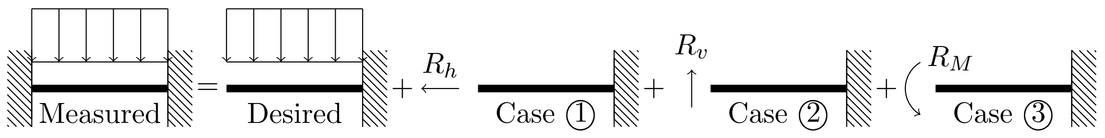


Figure C.1. The superimposed cases considered in this appendix. A generic horizontal beam is shown, supported on one or both sides. In each case, the beam is loaded by an arbitrary distributed load, a horizontal force R_h, a vertical force R_v, or a moment R_M.

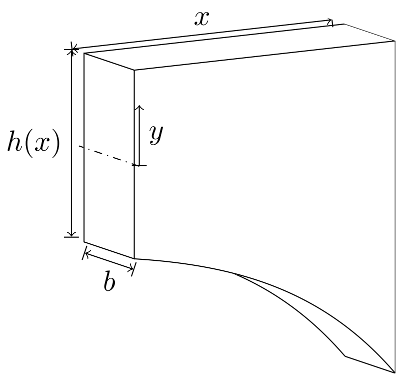


Figure C.2. A cross-section through a cantilever of continuously-varying height a distance x from the fixed support on the right. The neutral axis is assumed to be at the centre of the cross-section, as indicated by the dash-dotted line.

The internal stress distributions are therefore then integrated over the areas comprising the links on the top and bottom of the structure to produce a resultant moment $M_{ⓘ,\eta}$ and axial force $F_{ⓘ,\eta}$, each defined $\{\forall\ i \in \{1,2,3\} \wedge \eta \in \{\mathit{top}, \mathit{bottom}\}\}$. The resultant $M_{ⓘ,\eta}$ and $F_{ⓘ,\eta}$ are subtracted from the equivalent measured values to approximate the values of M and F in the links at the top and bottom of the structure in the desired case. These values are used by the active agent in the deconstruction algorithm to determine a course of action when it leaves the `gathering` mode.

C.1 Case ①: Horizontal Reaction Force

The horizontal reaction force R_h produces the simplest longitudinal stress distribution. This case is illustrated in Figure C.3. The force R_h is assumed to be spread evenly over the whole cross-section, so the longitudinal stress is given by:

$$\sigma_{①} = \frac{R_h}{bh(x)} \tag{C.3}$$

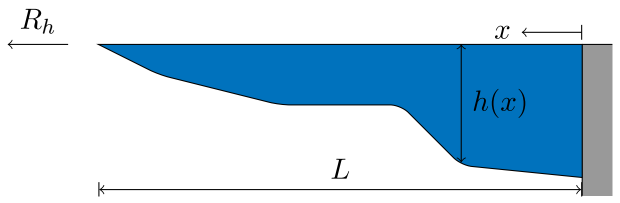


Figure C.3. A 2D representation of case ①. The cantilever is shown in blue, and the fixed support in grey.

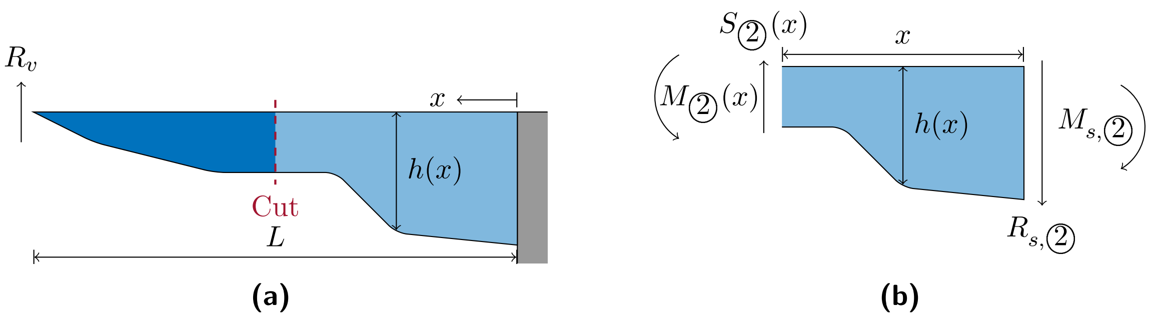


Figure C.4. A 2D representation of case ②. The cantilever is shown in blue, and the fixed support in grey. **(a)** The whole structure, and **(b)** a free body diagram of a cut made a distance x from the right support to reveal the internal shear force $S_{\textcircled{2}}(x)$ and moment $M_{\textcircled{2}}(x)$.

C.1.1 Axial Forces

To obtain the axial force, (C.3) is multiplied by the area of each link A_{link} (C.2), resulting in:

$$F_{\textcircled{1},top} = F_{\textcircled{1},bottom} = \frac{lR_h}{h(x)} \tag{C.4}$$

C.1.2 Moments

The longitudinal stress distribution is independent of y, so no resultant moment is produced in any links:

$$M_{\textcircled{1},top} = M_{\textcircled{1},bottom} = 0 \tag{C.5}$$

C.2 Case ②: Vertical Reaction Force

Figure C.4 shows the case for the vertical reaction force R_v. A cut is made a distance x from the fixed support to reveal the internal shear force $S_{\textcircled{2}}(x)$ and moment $M_{\textcircled{2}}(x)$. The reaction force and moment at the support are denoted $R_{s,\textcircled{2}}$ and $M_{s,\textcircled{2}}$ respectively. Vertical force equilibrium for the complete and cut structures yields:

$$S_{\textcircled{2}}(x) = R_{s,\textcircled{2}} = R_v \tag{C.6}$$

Moment equilibrium of the whole structure is then taken to give:

$$M_{s,\textcircled{2}} = -R_v L \tag{C.7}$$

It is now possible to calculate the internal moment by considering moment equilibrium of the cut structure:

$$\begin{aligned}
M_{\textcircled{2}}(x) &= R_{s,\textcircled{2}} x + M_{s,\textcircled{2}} \\
M_{\textcircled{2}}(x) &= R_v x - R_v L \\
M_{\textcircled{2}}(x) &= R_v(x - L)
\end{aligned} \tag{C.8}$$

The internal moment along the structure allows the calculation of the longitudinal stress distribution using the standard equation for a linear elastic beam [188]:

$$\begin{aligned}
\sigma_{\textcircled{2}}(x, y) &= \frac{M_{\textcircled{2}}(x) y}{I} \\
\sigma_{\textcircled{2}}(x, y) &= \frac{R_v(x - L) \cdot y}{\frac{bh(x)^3}{12}} \\
\sigma_{\textcircled{2}}(x, y) &= \frac{12 R_v y (x - L)}{bh(x)^3}
\end{aligned} \tag{C.9}$$

The longitudinal stress distribution across the cross-section at x is shown in Figure C.5. In order to calculate the resultant axial force f at different points in the cross-section, this distribution is integrated between two arbitrary points y_1 and y_2 and multiplying by the breadth b:

$$\begin{aligned}
f_{\textcircled{2}}(x, y_1, y_2) &= \int_{y_1}^{y_2} \sigma_{\textcircled{2}}(x, y) dy \cdot b \\
f_{\textcircled{2}}(x, y_1, y_2) &= \int_{y_1}^{y_2} \frac{12 R_v y (x - L)}{bh(x)^3} dy \cdot b \\
f_{\textcircled{2}}(x, y_1, y_2) &= \frac{12 R_v (x - L)}{h(x)^3} \left[\frac{y^2}{2} \right]_{y_1}^{y_2} \\
f_{\textcircled{2}}(x, y_1, y_2) &= \frac{6 R_v (x - L)}{h(x)^3} (y_2^2 - y_1^2)
\end{aligned} \tag{C.10}$$

C.2.1 Axial Forces

The axial force in the link on the top is calculated by evaluating (C.10) with $y_1 = \frac{h(x)}{2} - l$ and $y_2 = \frac{h(x)}{2}$. This gives:

$$\begin{aligned}
F_{\textcircled{2},top} &= \frac{6 R_v (x - L)}{h(x)^3} \left(\left(\frac{h(x)}{2} \right)^2 - \left(\frac{h(x)}{2} - l \right)^2 \right) \\
F_{\textcircled{2},top} &= \frac{6 R_v l}{h(x)^3} (x - L) \left(h(x) - l \right)
\end{aligned} \tag{C.11}$$

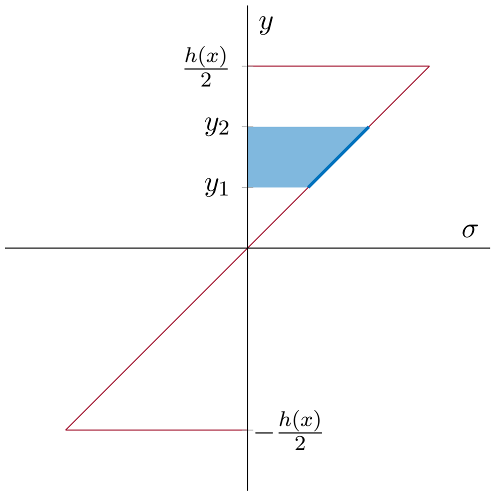


Figure C.5. The shape of the longitudinal stress distribution σ across the cross-section at x. The blue region is integrated to calculate the resultant axial force.

The longitudinal stress distribution is antisymmetric in the line $y = 0$, so the axial force in the bottom link is equal and opposite to that in the top link:

$$\begin{aligned}F_{\textcircled{2},bottom} &= -F_{\textcircled{2},top} \\ F_{\textcircled{2},bottom} &= -\frac{6R_v l}{h(x)^3}(x-L)\left(h(x)-l\right) \qquad \text{(C.12)}\end{aligned}$$

C.2.2 Moments

Calculating the moment about the centre of the top link requires integrating two regions to find the forces in this link above and below its centre, $f_{\textcircled{2},1}$ and $f_{\textcircled{2},2}$. These regions are shown in Figure C.6. The forces are found by substituting appropriate y_1 and y_2 into (C.10):

$$\begin{aligned}f_{\textcircled{2},1} &= \frac{6R_v(x-L)}{h(x)^3}\left(\left(\frac{h(x)}{2}\right)^2 - \left(\frac{h(x)}{2}-\frac{l}{2}\right)^2\right) \\ f_{\textcircled{2},1} &= \frac{3R_v l(x-L)}{h(x)^3}\left(h(x)-\frac{l}{2}\right) \qquad \text{(C.13)}\end{aligned}$$

$$\begin{aligned}f_{\textcircled{2},2} &= \frac{6R_v(x-L)}{h(x)^3}\left(\left(\frac{h(x)}{2}-\frac{l}{2}\right)^2 - \left(\frac{h(x)}{2}-l\right)^2\right) \\ f_{\textcircled{2},2} &= \frac{3R_v l(x-L)}{h(x)^3}\left(h(x)-\frac{3l}{2}\right) \qquad \text{(C.14)}\end{aligned}$$

These forces act at the centroid of these regions. These centroids are at distances $\bar{y}_{\textcircled{2},1}$ and $\bar{y}_{\textcircled{2},2}$ from the neutral axis, assumed to be at the centre of the cross-section.

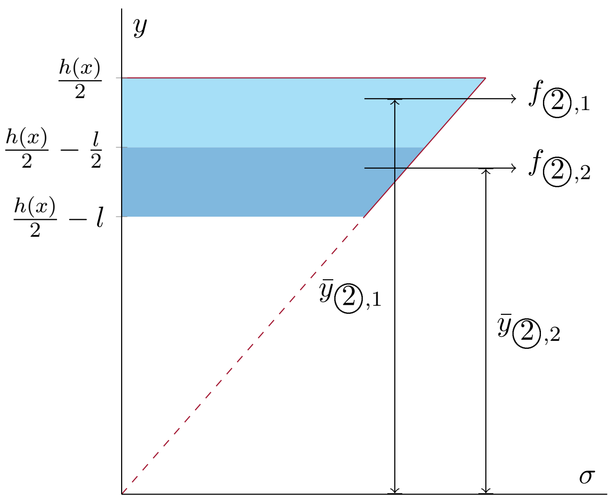


Figure C.6. The regions of the longitudinal stress distribution in the cross-section at x used to calculate the moment in the top link. The forces in the two regions $f_{\textcircled{2},1}$ and $f_{\textcircled{2},2}$ act at distances $\bar{y}_{\textcircled{2},1}$ and $\bar{y}_{\textcircled{2},2}$ from the neutral axis in the centre of the cross-section.

For an arbitrary region n bounded by y_1 and y_2:

$$\bar{y}_{\textcircled{2},n} = \frac{1}{\int_{y_1}^{y_2} \sigma_{\textcircled{2}}(x,y)dy} \int_{y_1}^{y_2} y\sigma_{\textcircled{2}}(x,y)dy$$

$$\bar{y}_{\textcircled{2},n} = \frac{1}{\frac{f_{\textcircled{2},n}}{b}} \int_{y_1}^{y_2} y \cdot \frac{12R_v y(x-L)}{bh(x)^3} dy$$

$$\bar{y}_{\textcircled{2},n} = \frac{12R_v(x-L)}{f_{\textcircled{2},n}h(x)^3} \left[\frac{y^3}{3}\right]_{y_1}^{y_2}$$

$$\bar{y}_{\textcircled{2},n} = \frac{4R_v(x-L)}{f_{\textcircled{2},n}h(x)^3} \left(y_2^3 - y_1^3\right) \tag{C.15}$$

This equation gives the following distances:

$$\bar{y}_{\textcircled{2},1} = \frac{4R_v(x-L)}{f_{\textcircled{2},1}h(x)^3} \left(\left(\frac{h(x)}{2}\right)^3 - \left(\frac{h(x)}{2} - \frac{l}{2}\right)^3\right)$$

$$\bar{y}_{\textcircled{2},1} = \frac{R_v(x-L)}{2f_{\textcircled{2},1}h(x)^3} \left(3lh(x)^2 - 3l^2h(x) + l^3\right) \tag{C.16}$$

$$\bar{y}_{\textcircled{2},2} = \frac{4R_v(x-L)}{f_{\textcircled{2},2}h(x)^3} \left(\left(\frac{h(x)}{2} - \frac{l}{2}\right)^3 - \left(\frac{h(x)}{2} - l\right)^3\right)$$

$$\bar{y}_{\textcircled{2},2} = \frac{R_v(x-L)}{2f_{\textcircled{2},2}h(x)^3} \left(3lh(x)^2 - 9l^2h(x) + 7l^3\right) \tag{C.17}$$

The moment can then be calculated by combining (C.13), (C.14), (C.16), and (C.17). These values are combined as follows:

$$M_{\textcircled{2},top} = f_{\textcircled{2},2} \cdot \left(\left(\frac{h}{2} - \frac{l}{2}\right) - \bar{y}_{\textcircled{2},2}\right) - f_{\textcircled{2},1} \cdot \left(\bar{y}_{\textcircled{2},1} - \left(\frac{h}{2} - \frac{l}{2}\right)\right) \tag{C.18}$$

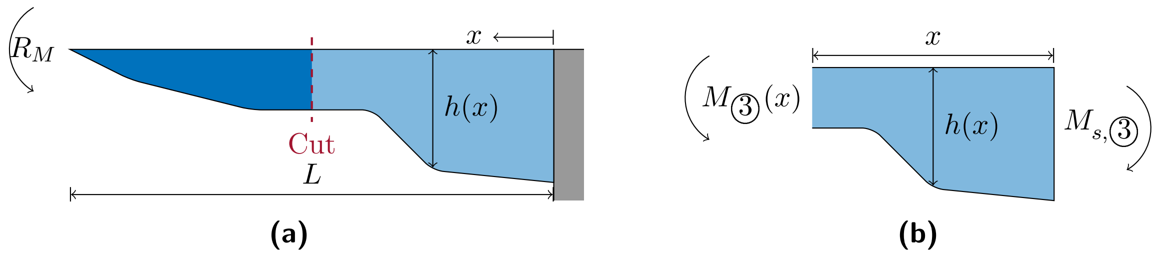


Figure C.7. A 2D representation of case ③. The cantilever is shown in blue, and the fixed support in grey. **(a)** The whole structure, and **(b)** a free body diagram of a cut made a distance x from the right support to reveal the internal moment $M_{\textcircled{3}}(x)$.

The moment in the bottom link will be equal and in the same direction to that at the top:

$$
\begin{aligned}
M_{\textcircled{2},bottom} &= M_{\textcircled{2},top} \\
M_{\textcircled{2},bottom} &= f_{\textcircled{2},2} \cdot \left(\left(\frac{h}{2} - \frac{l}{2} \right) - \bar{y}_{\textcircled{2},2} \right) - f_{\textcircled{2},1} \cdot \left(\bar{y}_{\textcircled{2},1} - \left(\frac{h}{2} - \frac{l}{2} \right) \right) \qquad \text{(C.19)}
\end{aligned}
$$

C.3 Case ③: Reaction Moment

The final cases considers the reaction moment R_M, illustrated in Figure C.4. By considering moment equilibrium of the whole structure, it can be seen that:

$$M_{\textcircled{3}}(x) = M_{s,\textcircled{3}} = R_M \qquad \text{(C.20)}$$

The longitudinal stress distribution is therefore the same as for the vertical reaction force in (C.9), but with $M_{\textcircled{2}}(x) = R_v(x - L)$ replaced by $M_{\textcircled{3}}(x) = R_M$ to yield:

$$\sigma_{\textcircled{3}}(x, y) = \frac{12 R_M y}{b h(x)^3} \qquad \text{(C.21)}$$

C.3.1 Axial Forces

Continuing the same analysis as in Section C.2.1 but with $M_{\textcircled{2}}(x) = R_v(x - L)$ replaced by $M_{\textcircled{3}}(x) = R_M$ gives the axial force in the links on the top and bottom of the structure as:

$$F_{\textcircled{3},top} = \frac{6 R_M l}{h(x)^3} \left(h(x) - l \right) \qquad \text{(C.22)}$$

$$F_{\textcircled{3},bottom} = -\frac{6 R_M l}{h(x)^3} \left(h(x) - l \right) \qquad \text{(C.23)}$$

C.3.2 Moments

The moments are calculated in the same manner as Section C.2.2, but again substituting $M_{\textcircled{2}}(x) = R_v(x - L)$ for $M_{\textcircled{3}}(x) = R_M$. This means the moments in the top and bottom links are given by:

$$M_{\textcircled{3},top} = f_{\textcircled{3},2} \cdot \left(\left(\frac{h}{2} - \frac{l}{2} \right) - \bar{y}_{\textcircled{3},2} \right) - f_{\textcircled{3},1} \cdot \left(\bar{y}_{\textcircled{3},1} - \left(\frac{h}{2} - \frac{l}{2} \right) \right) \tag{C.24}$$

$$M_{\textcircled{3},bottom} = M_{\textcircled{3},top} \tag{C.25}$$

where:

$$f_{\textcircled{3},1} = \frac{3R_M l}{h(x)^3} \left(h(x) - \frac{l}{2} \right) \tag{C.26}$$

$$f_{\textcircled{3},2} = \frac{3R_M l}{h(x)^3} \left(h(x) - \frac{3l}{2} \right) \tag{C.27}$$

$$\bar{y}_{\textcircled{3},1} = \frac{R_M}{2f_{\textcircled{3},1} h(x)^3} \left(3lh(x)^2 - 3l^2 h(x) + l^3 \right) \tag{C.28}$$

$$\bar{y}_{\textcircled{3},2} = \frac{R_M}{2f_{\textcircled{3},2} h(x)^3} \left(3lh(x)^2 - 9l^2 h(x) + 7l^3 \right) \tag{C.29}$$

Appendix D

Complete Bridge Optimisation Results

D.1 Varying $P_{release}(0)$

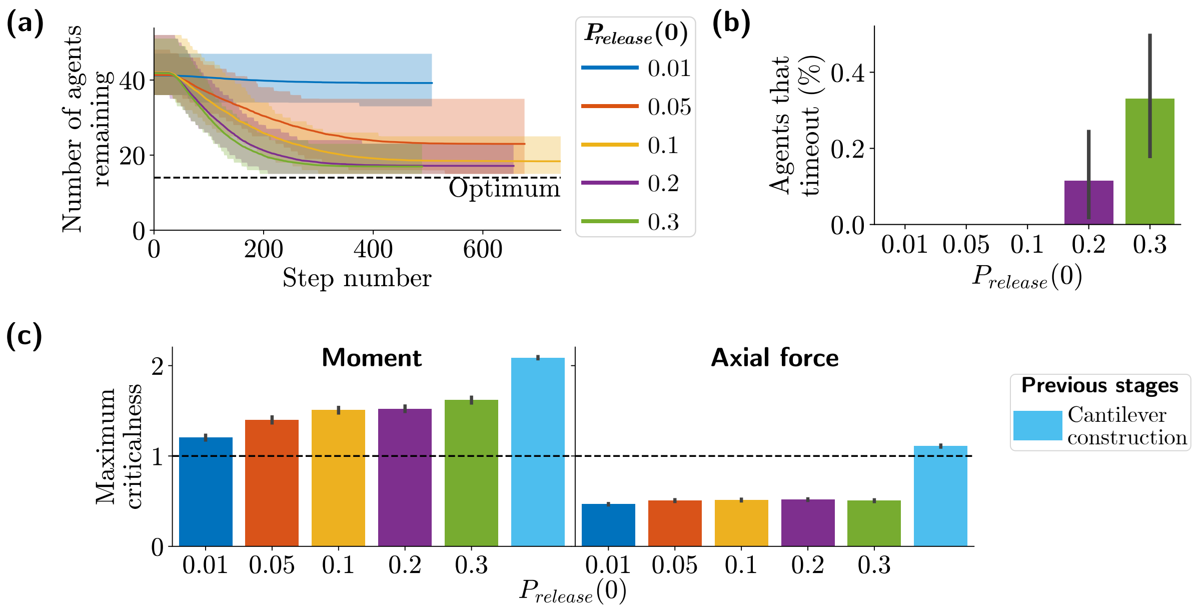


Figure D.1. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.2 m with weak links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

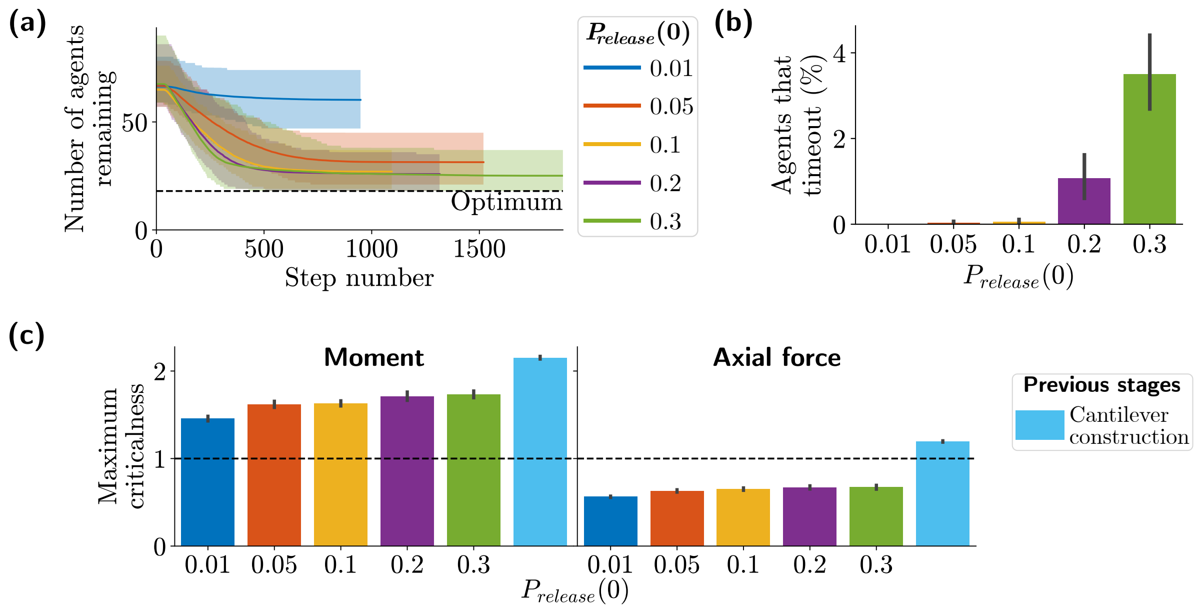


Figure D.2. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.4 m with weak links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

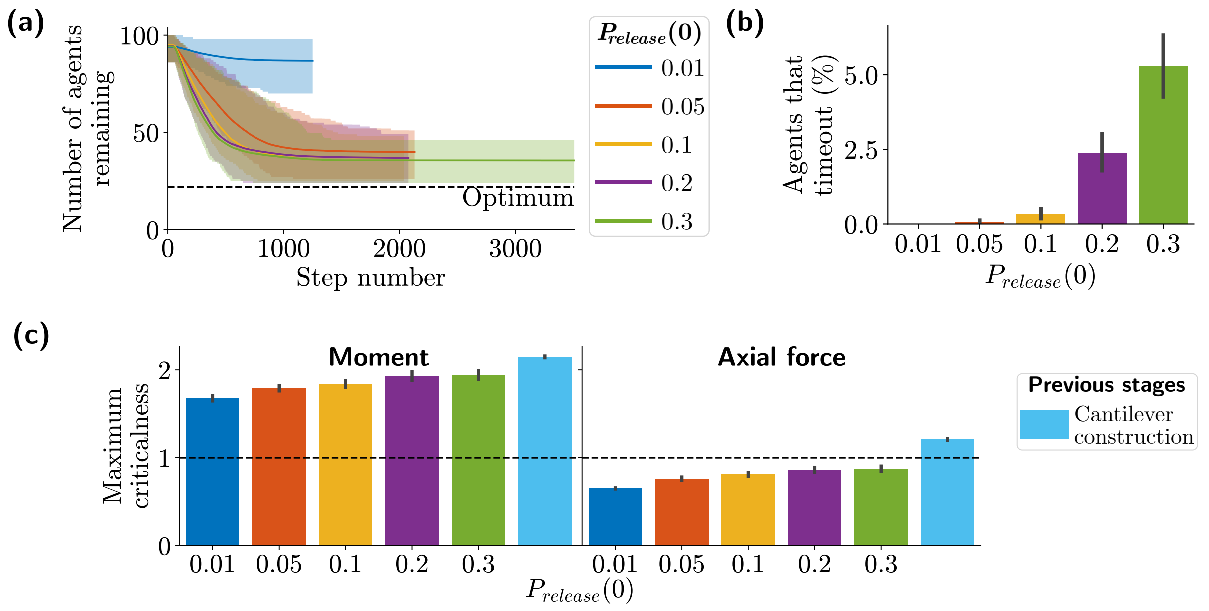


Figure D.3. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.6 m with weak links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

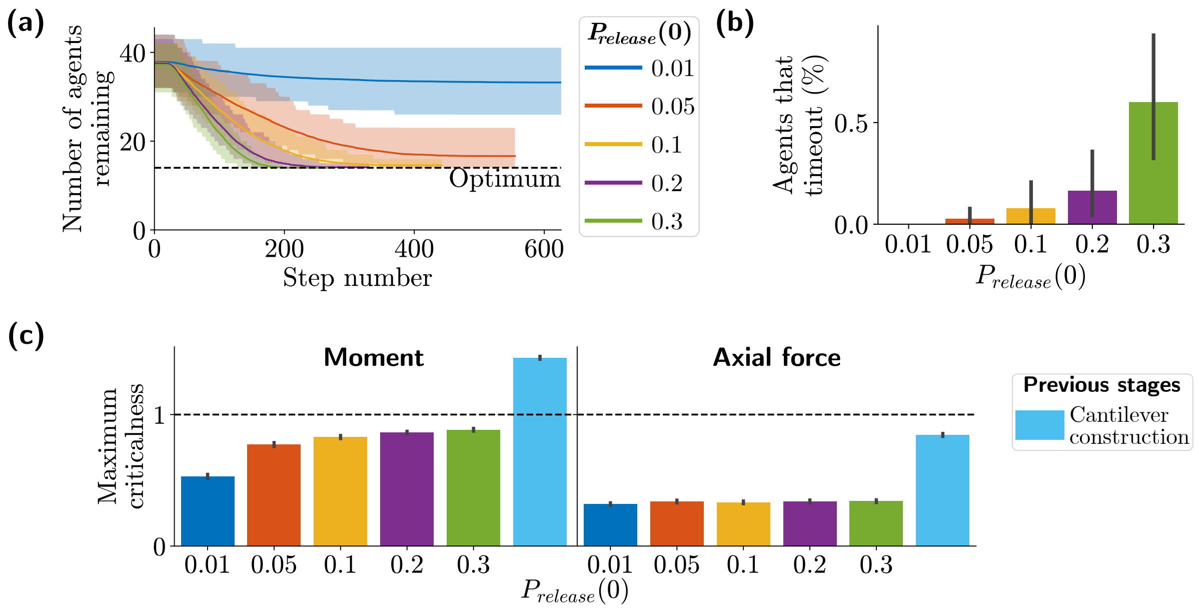


Figure D.4. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.4 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

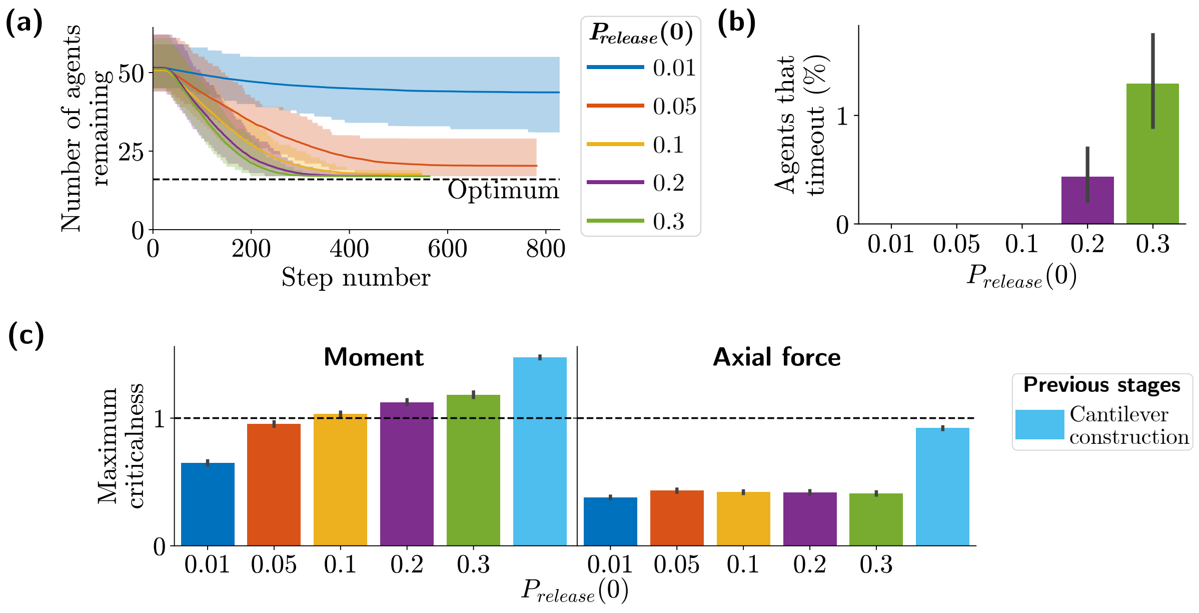


Figure D.5. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.6 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

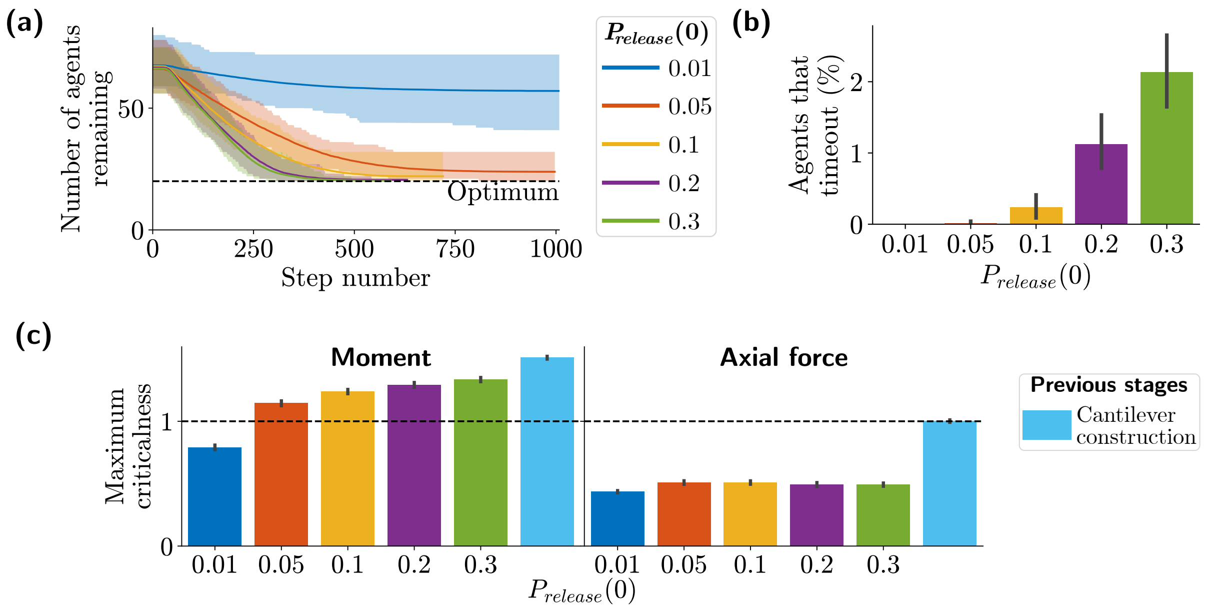


Figure D.6. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.8 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

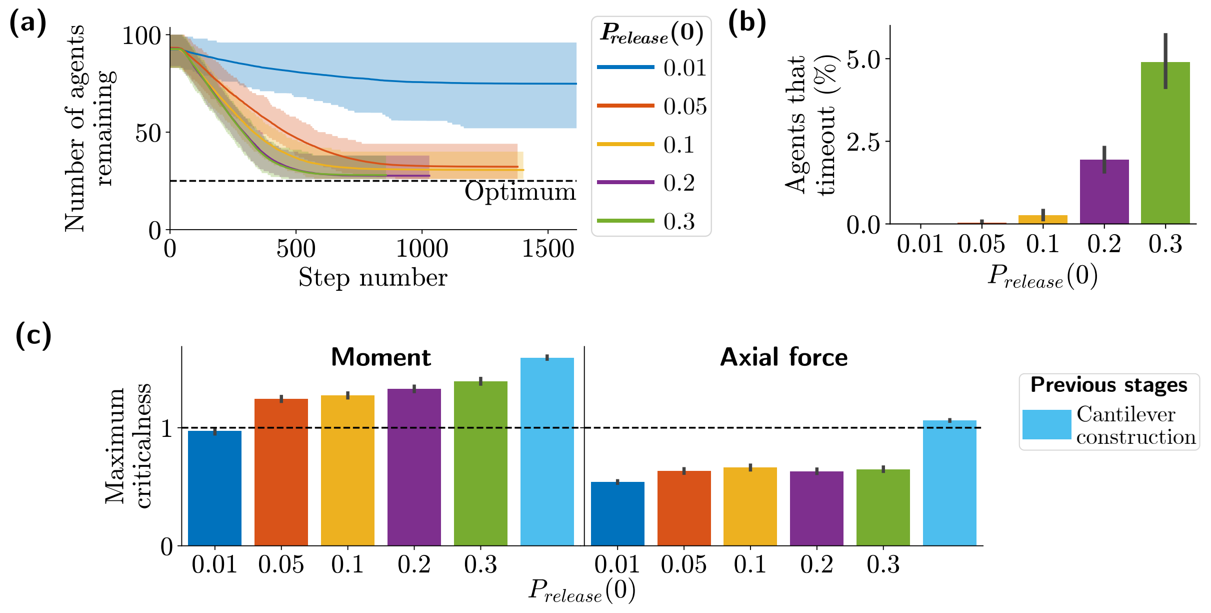


Figure D.7. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 2.1 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

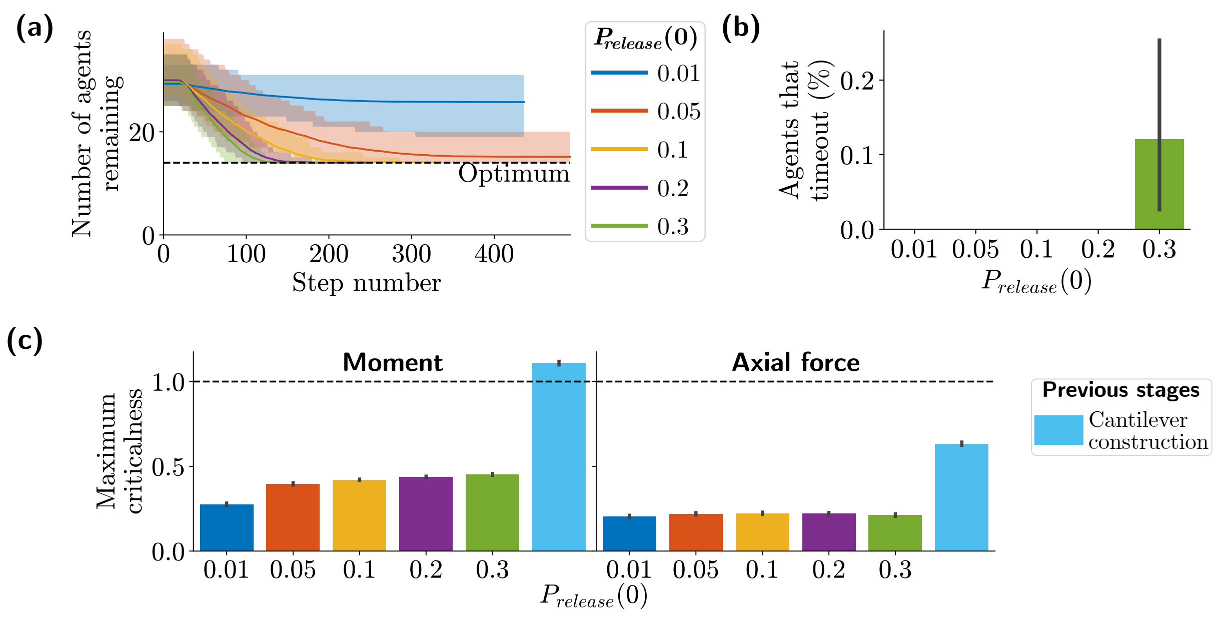


Figure D.8. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.4 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

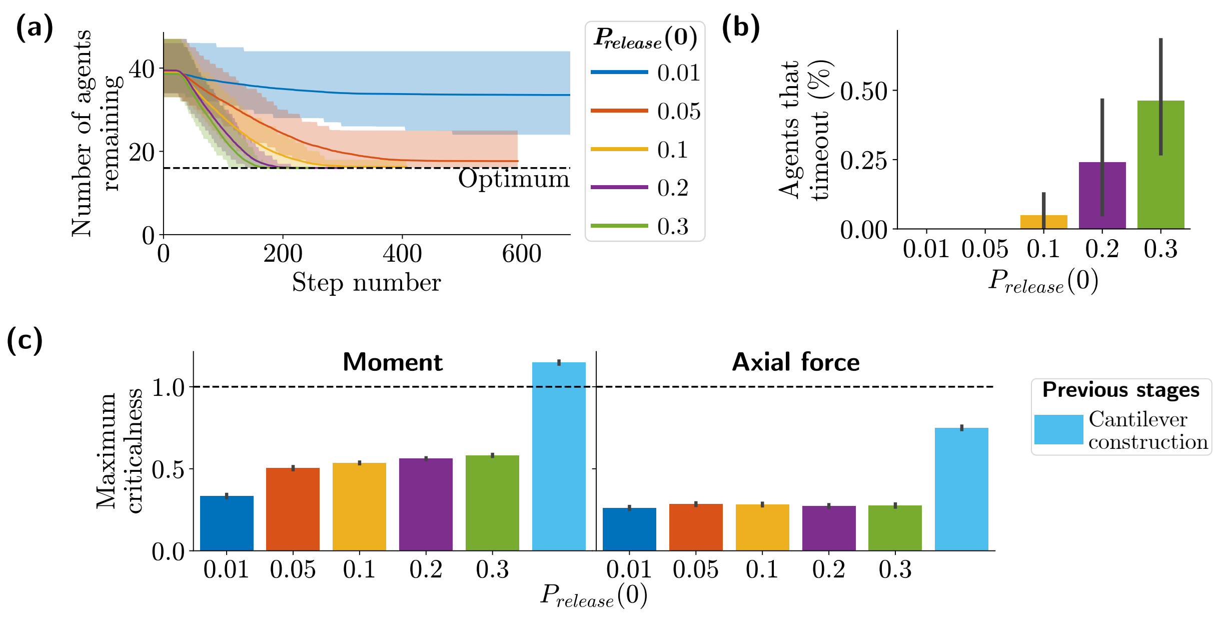


Figure D.9. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.6 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

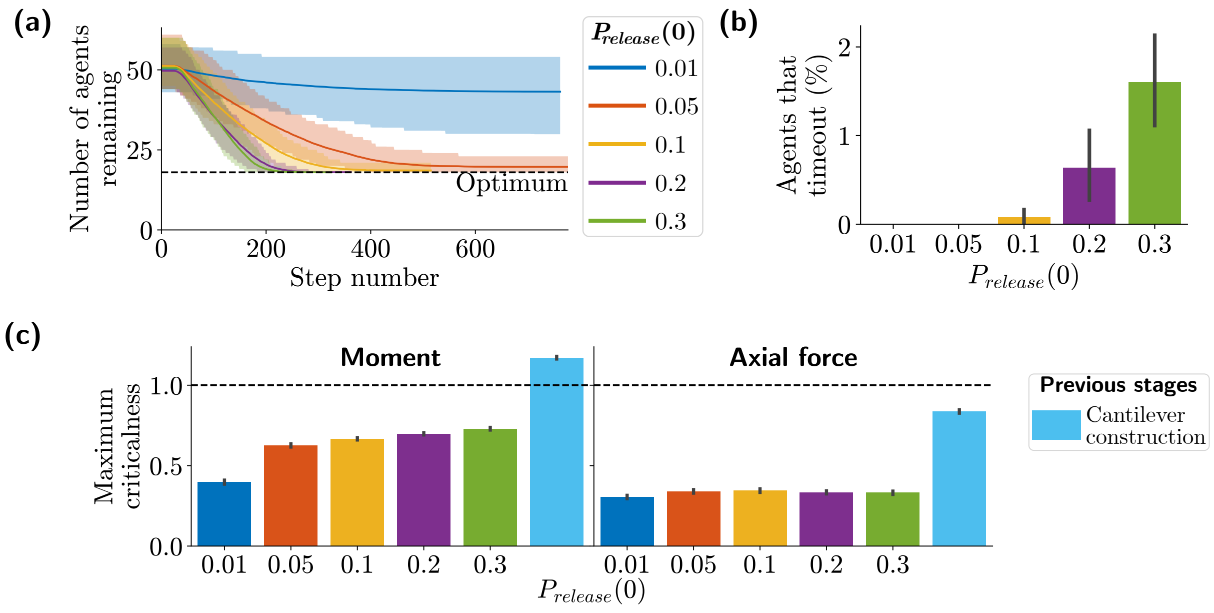


Figure D.10. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 1.8 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

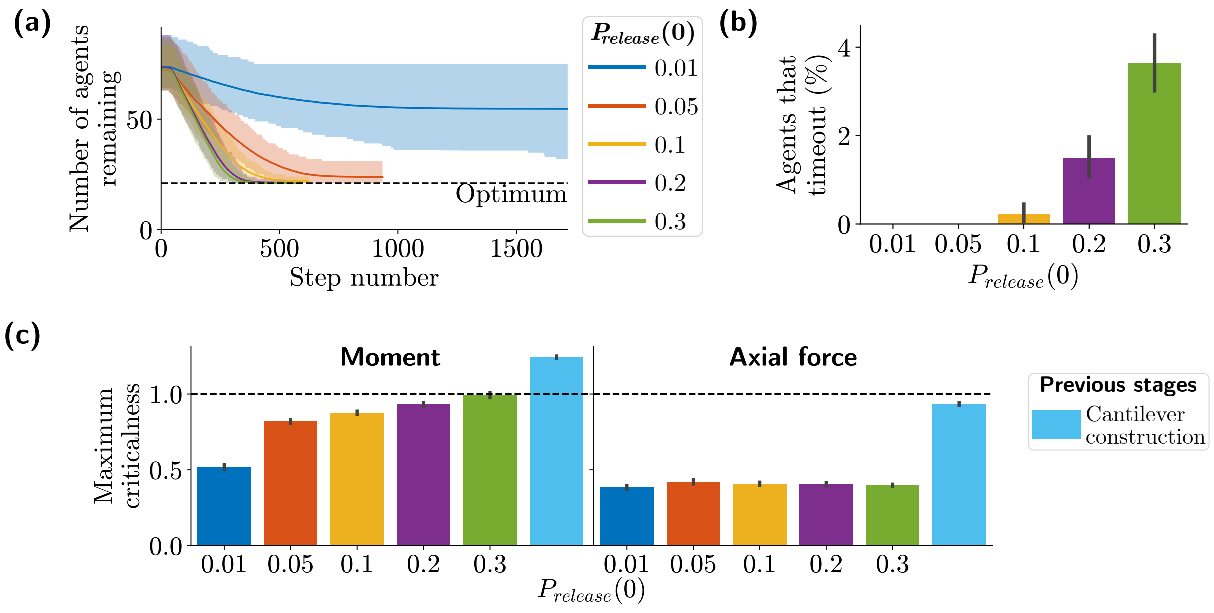


Figure D.11. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 2.1 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

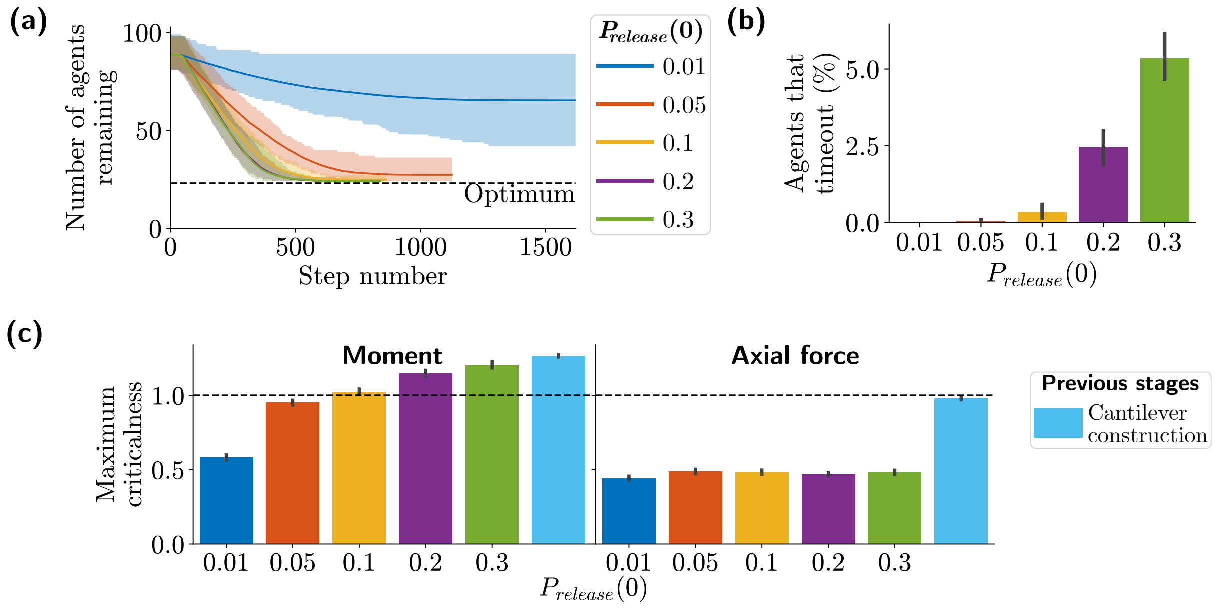
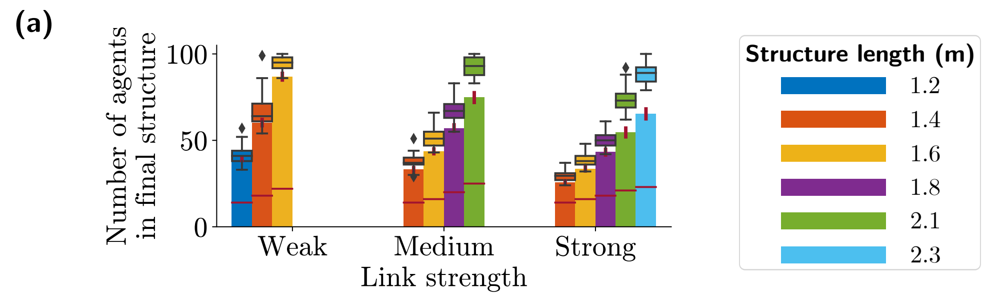


Figure D.12. The effect of varying $P_{release}(0)$ (denoted μ) on **(a)** the rate of agent removal, **(b)** the percentage of agents in the initial bridge that timeout, and **(c)** the maximum moment and axial force criticalness throughout the optimisation. The results are shown for 100 trials for bridges of length 2.3 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction is included for comparison.

D.2 Varying Structure Length



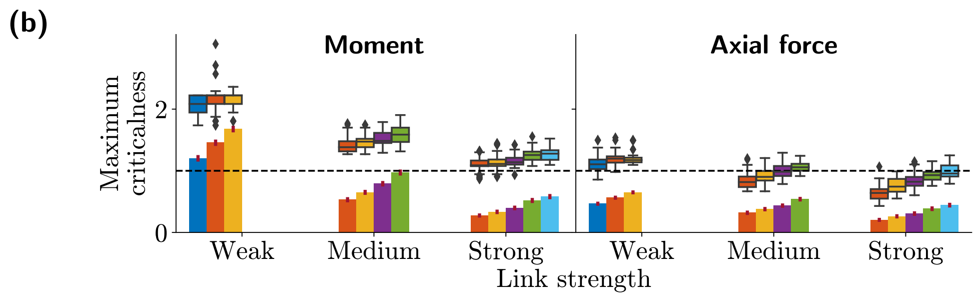


Figure D.13. The effect of varying structure length on **(a)** the number of agents remaining in the bridge after optimisation, and **(b)** the maximum moment and axial force criticalness throughout the optimisation. Each bar shows the mean of 100 trials for $\mu = 0.01$, vertical red lines show the 95 % confidence intervals, and boxplots show the equivalent data during construction of the cantilevers used as the starting configurations. In **(a)**, the optimum numbers of agents for each bridge are shown by the horizontal red lines.

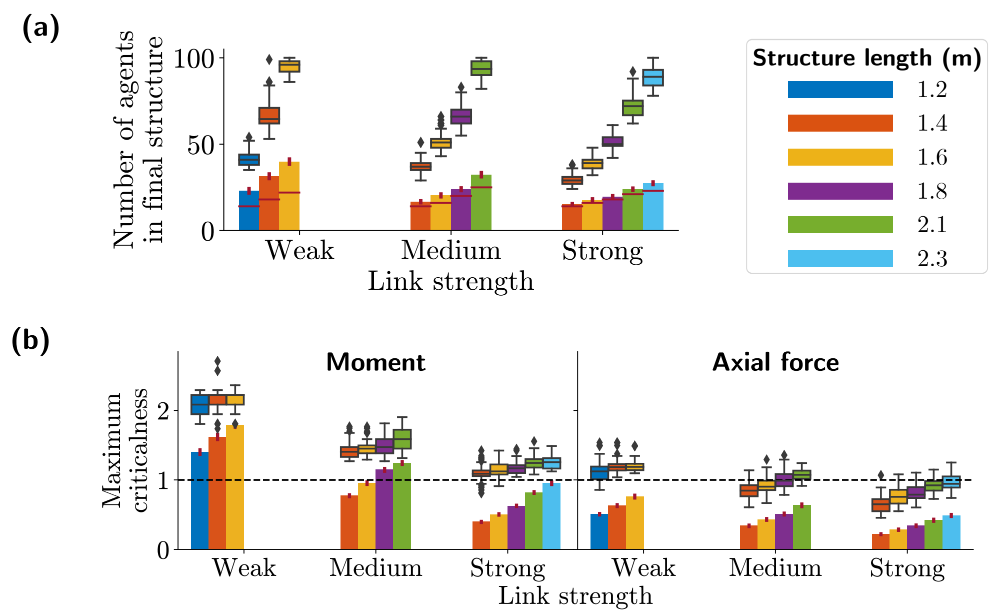


Figure D.14. The effect of varying structure length on **(a)** the number of agents remaining in the bridge after optimisation, and **(b)** the maximum moment and axial force criticalness throughout the optimisation. Each bar shows the mean of 100 trials for $\mu = 0.05$, vertical red lines show the 95 % confidence intervals, and boxplots show the equivalent data during construction of the cantilevers used as the starting configurations. In **(a)**, the optimum numbers of agents for each bridge are shown by the horizontal red lines.

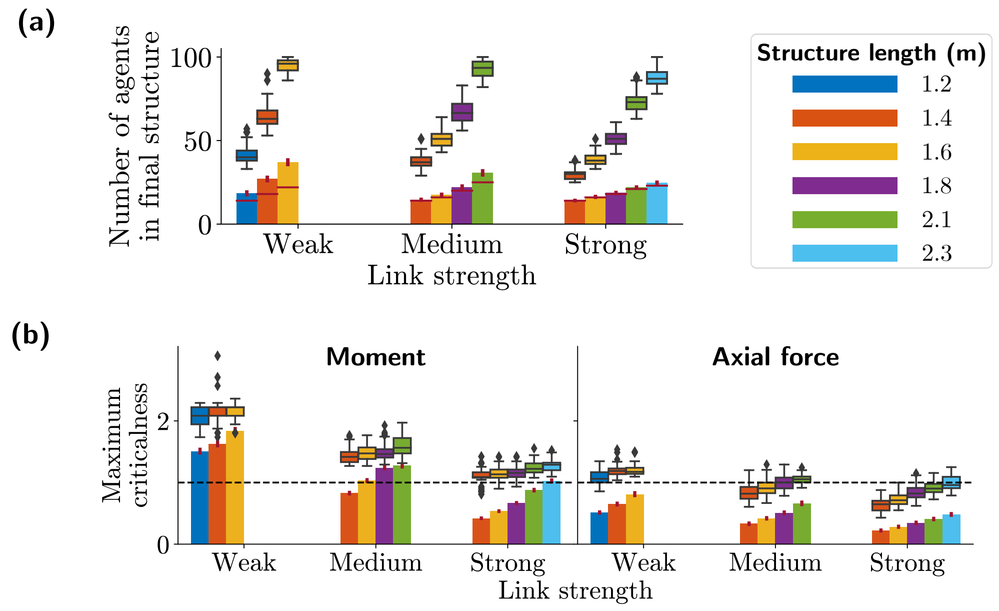


Figure D.15. The effect of varying structure length on **(a)** the number of agents remaining in the bridge after optimisation, and **(b)** the maximum moment and axial force criticalness throughout the optimisation. Each bar shows the mean of 100 trials for $\mu = 0.1$, vertical red lines show the 95 % confidence intervals, and boxplots show the equivalent data during construction of the cantilevers used as the starting configurations. In **(a)**, the optimum numbers of agents for each bridge are shown by the horizontal red lines.

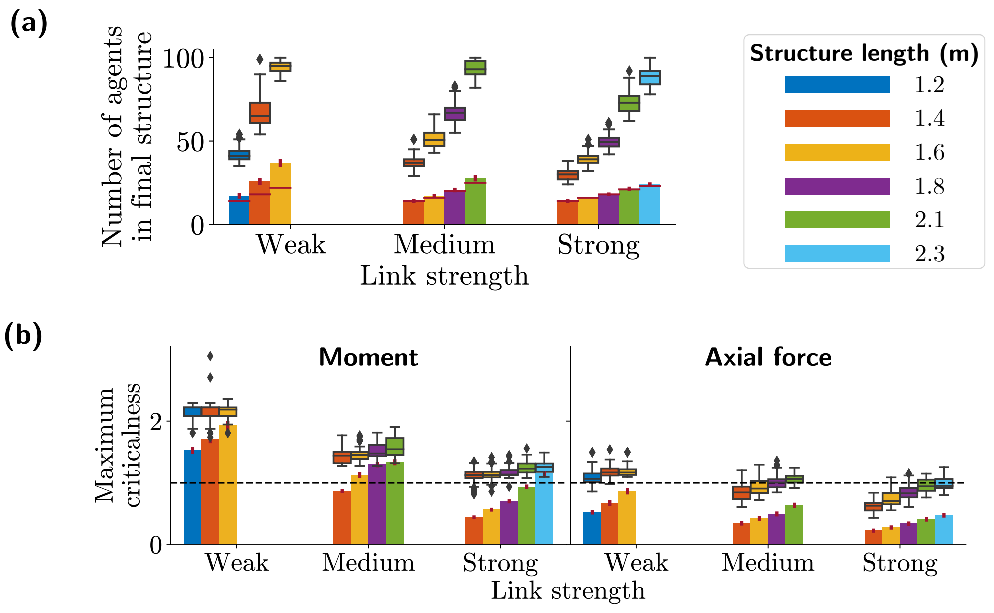


Figure D.16. The effect of varying structure length on **(a)** the number of agents remaining in the bridge after optimisation, and **(b)** the maximum moment and axial force criticalness throughout the optimisation. Each bar shows the mean of 100 trials for $\mu = 0.2$, vertical red lines show the 95 % confidence intervals, and boxplots show the equivalent data during construction of the cantilevers used as the starting configurations. In **(a)**, the optimum numbers of agents for each bridge are shown by the horizontal red lines.

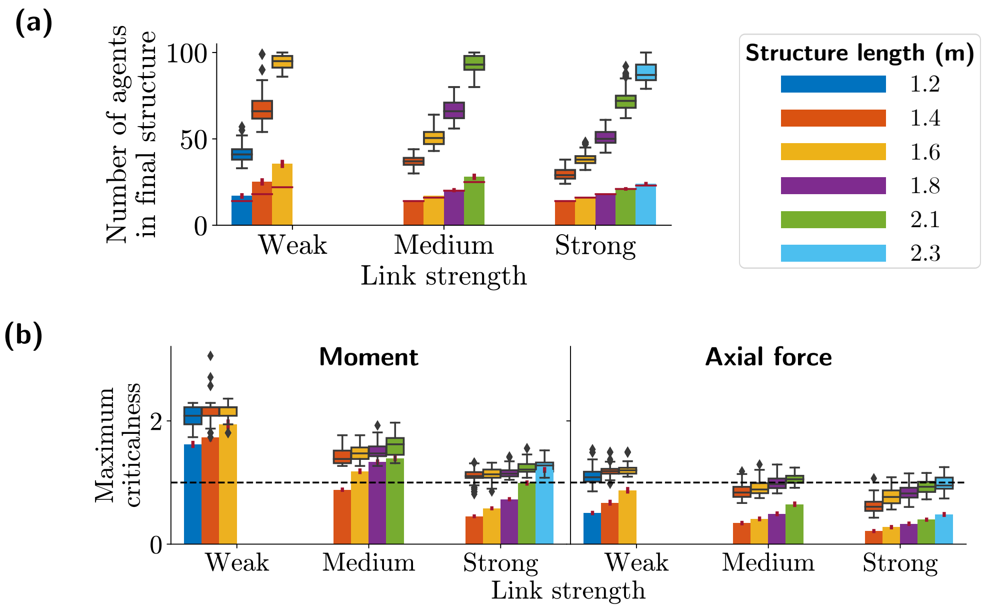


Figure D.17. The effect of varying structure length on **(a)** the number of agents remaining in the bridge after optimisation, and **(b)** the maximum moment and axial force criticalness throughout the optimisation. Each bar shows the mean of 100 trials for $\mu = 0.3$, vertical red lines show the 95 % confidence intervals, and boxplots show the equivalent data during construction of the cantilevers used as the starting configurations. In **(a)**, the optimum numbers of agents for each bridge are shown by the horizontal red lines.

Appendix E

Complete Bridge Deconstruction Results

E.1 Varying Agent Insertion Delay

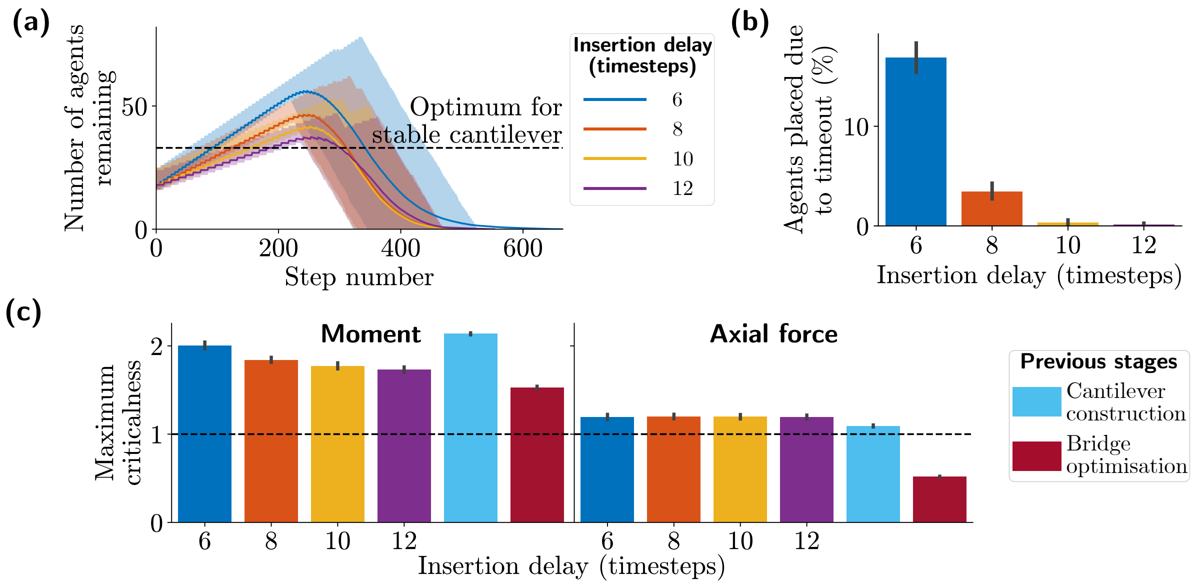


Figure E.1. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.2 m with weak links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

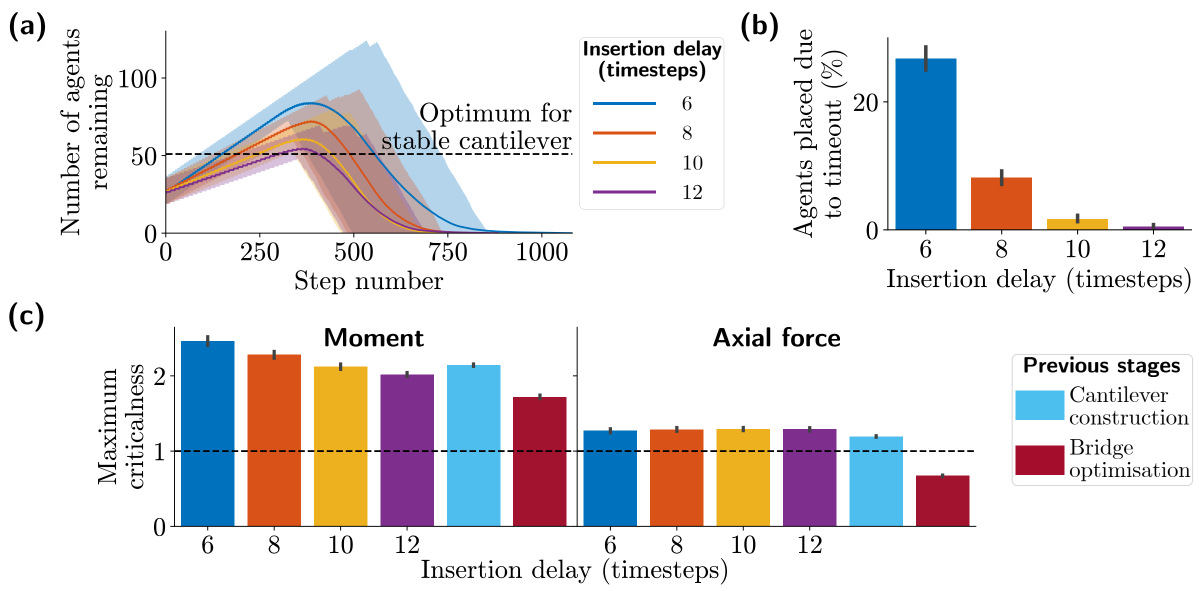


Figure E.2. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.4 m with weak links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

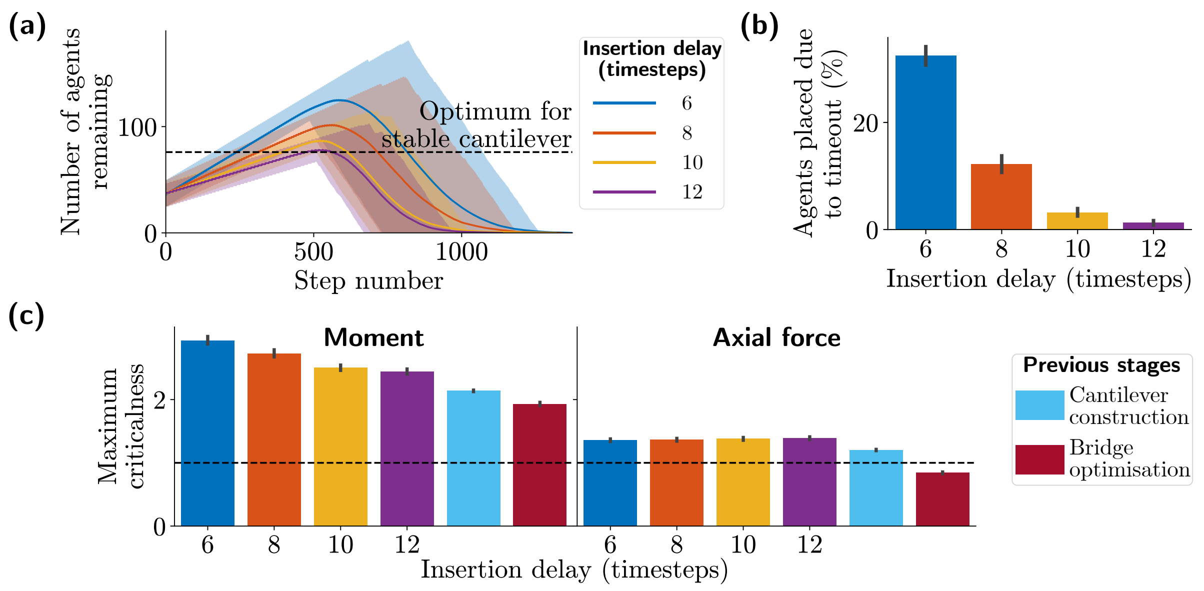


Figure E.3. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.6 m with weak links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

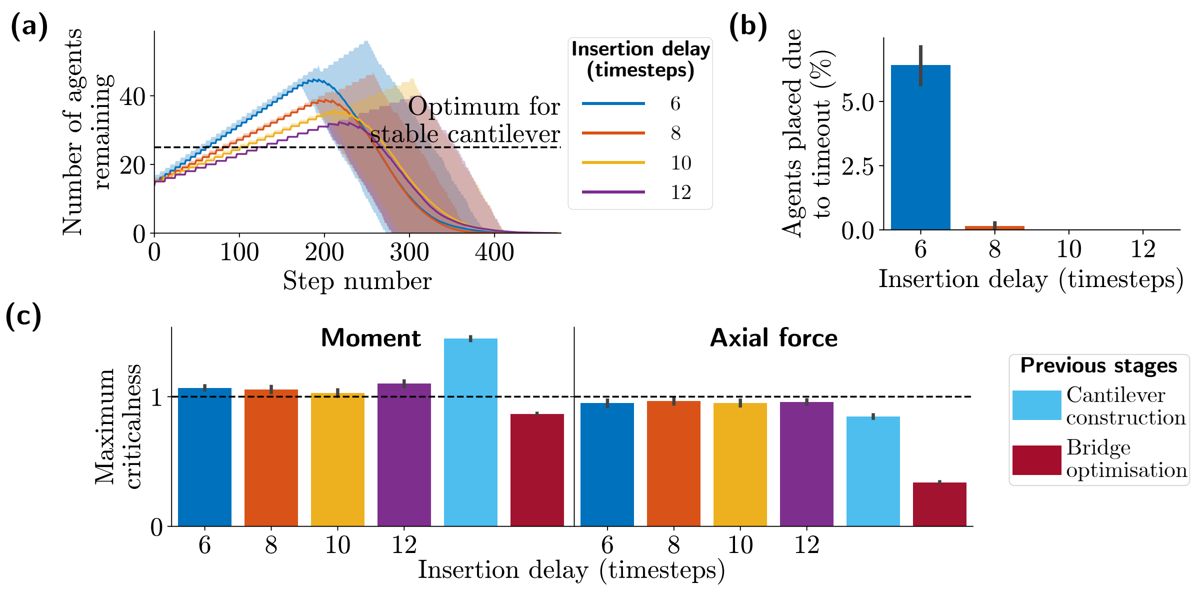


Figure E.4. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.4 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

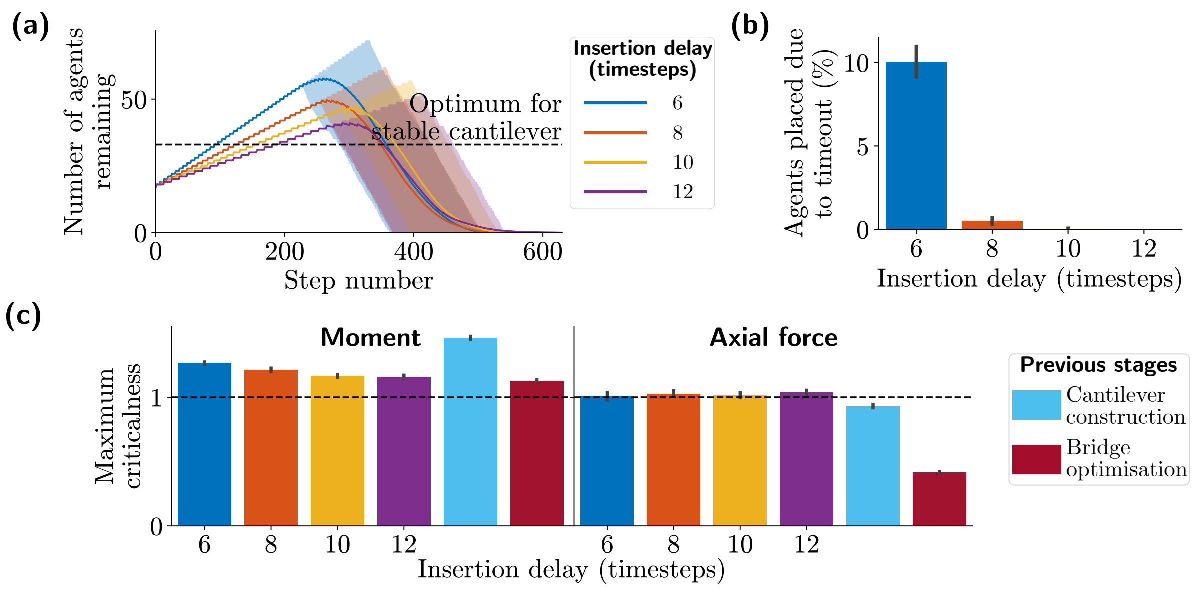


Figure E.5. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.6 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

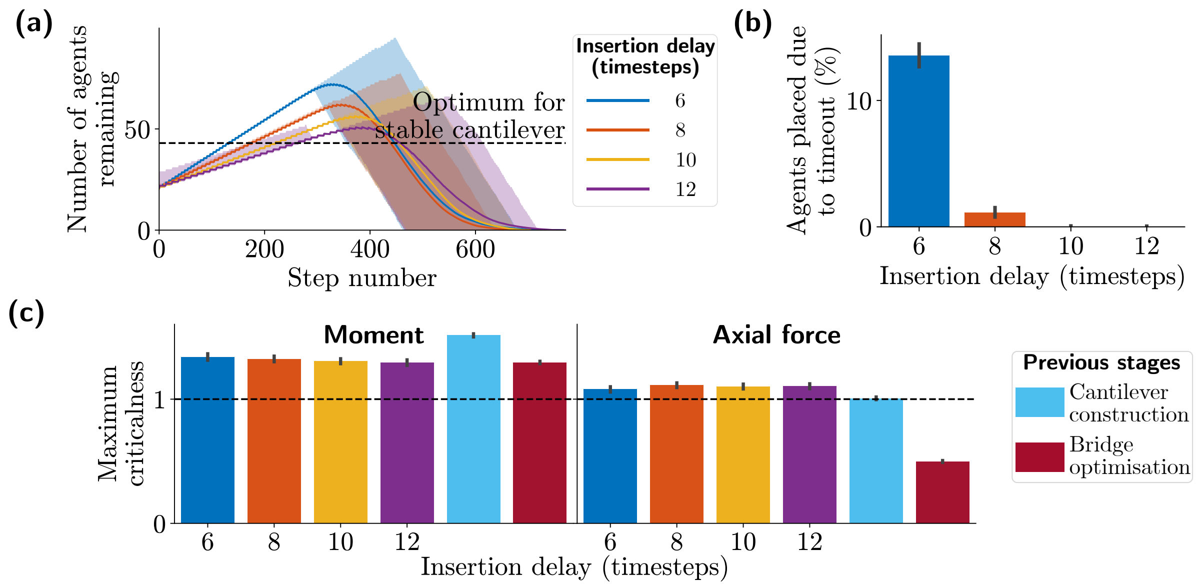


Figure E.6. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.8 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

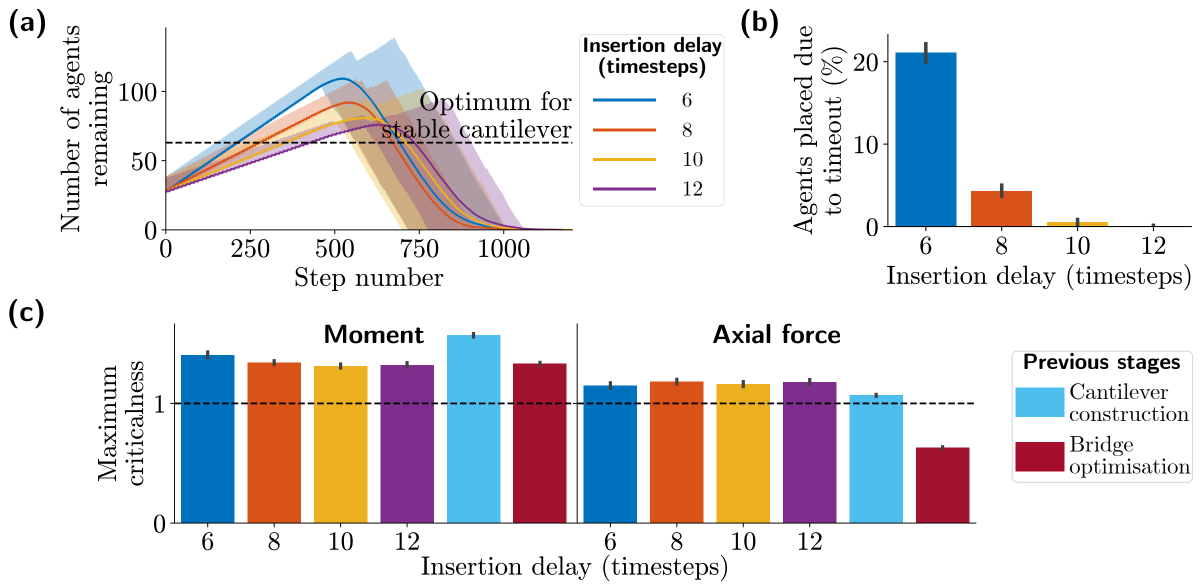


Figure E.7. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 2.1 m with links of medium strength. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

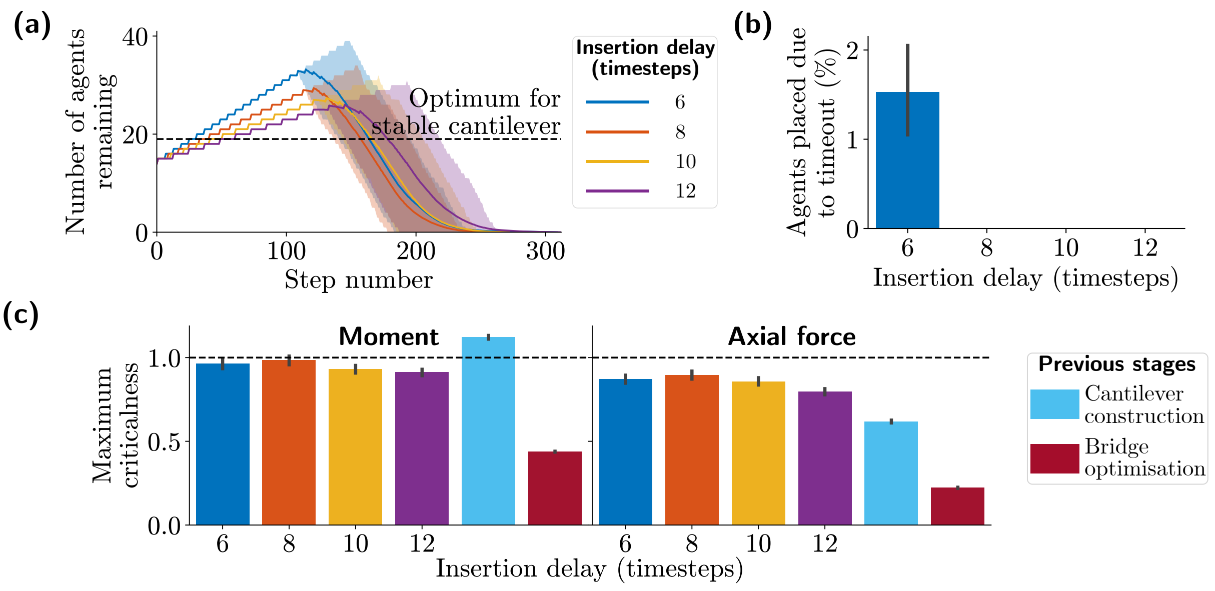


Figure E.8. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.4 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

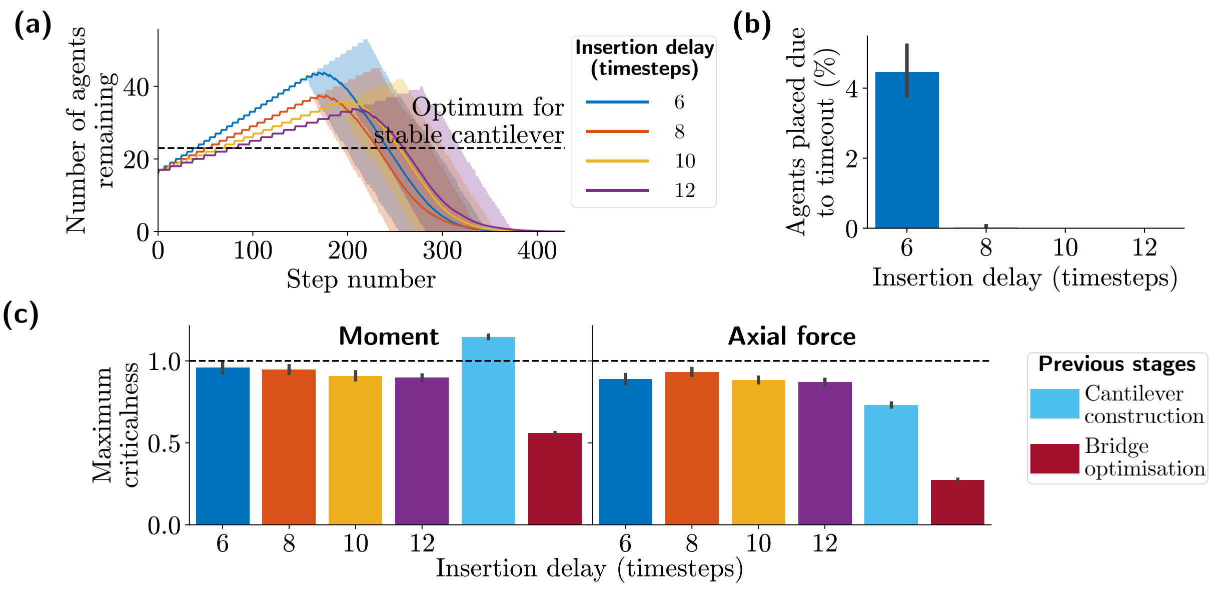


Figure E.9. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.6 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

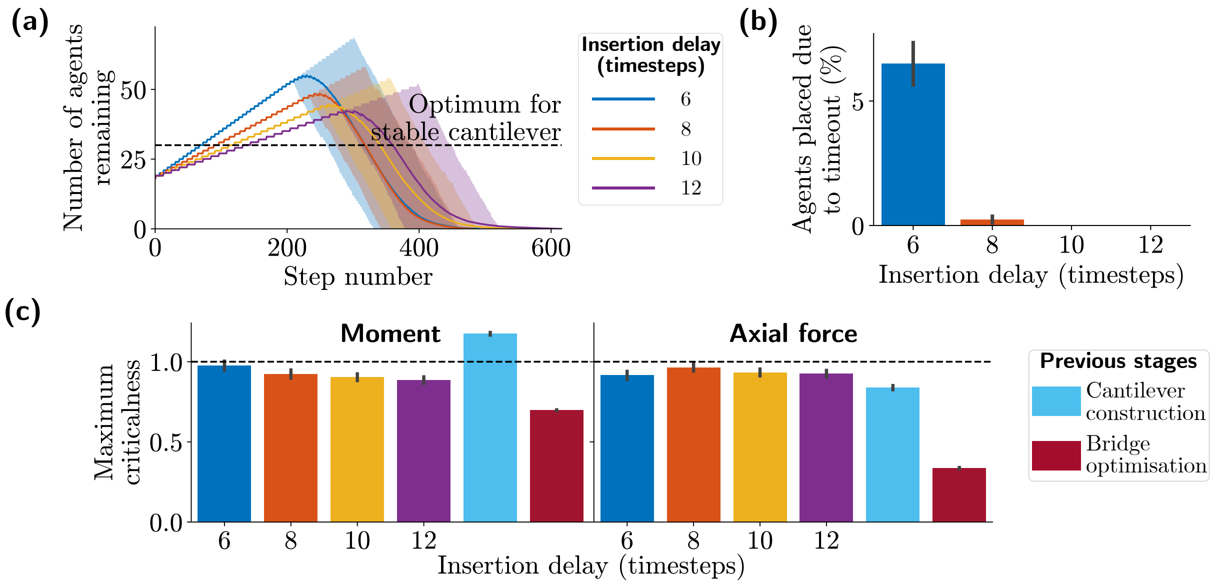


Figure E.10. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 1.8 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

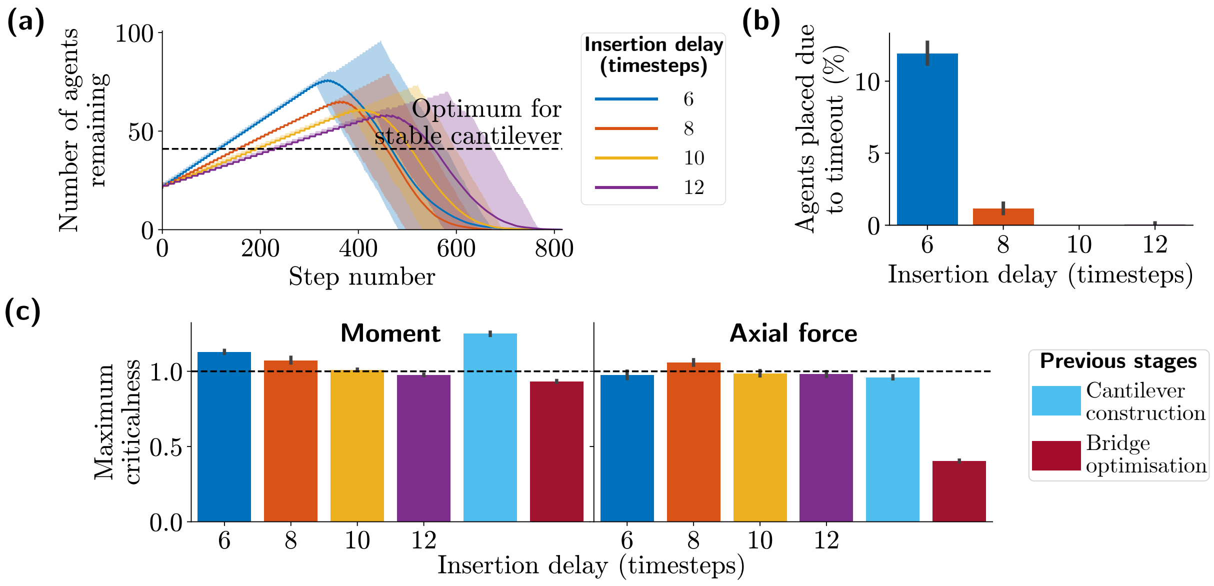


Figure E.11. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 2.1 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

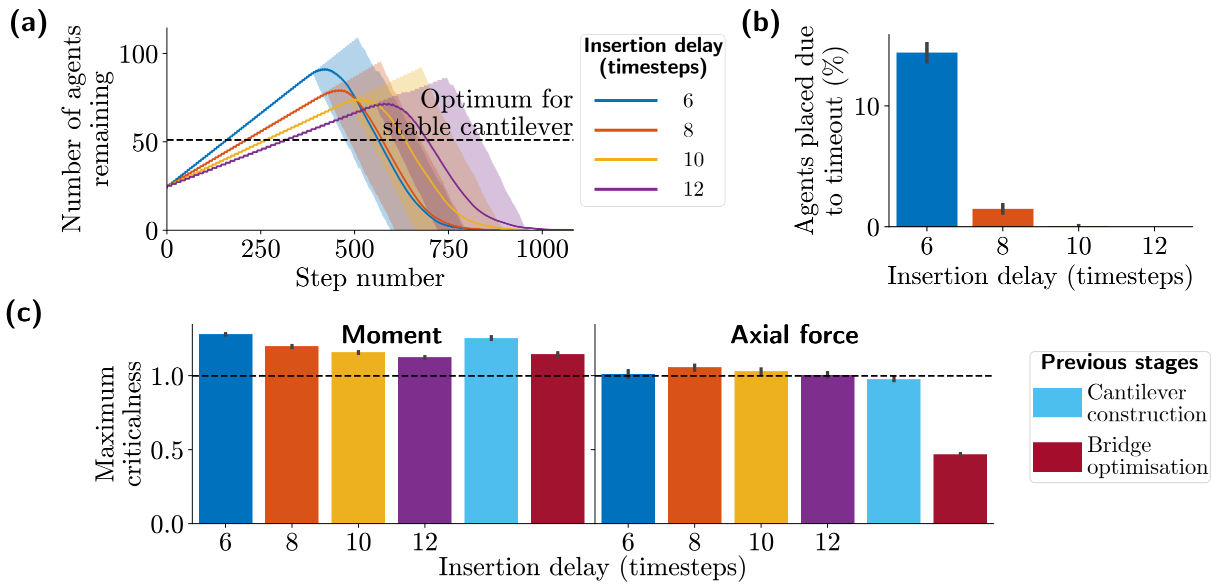
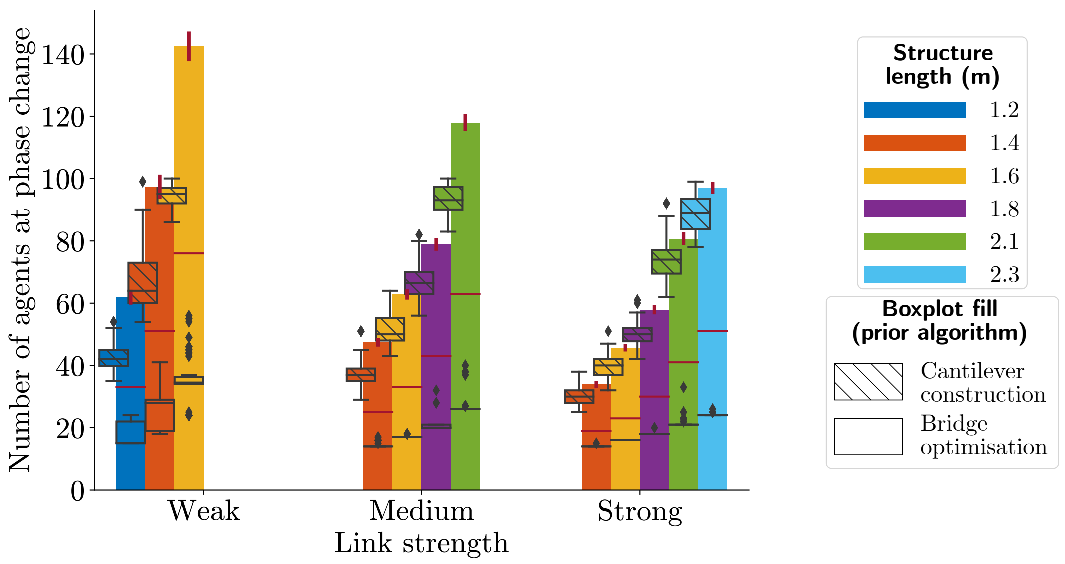
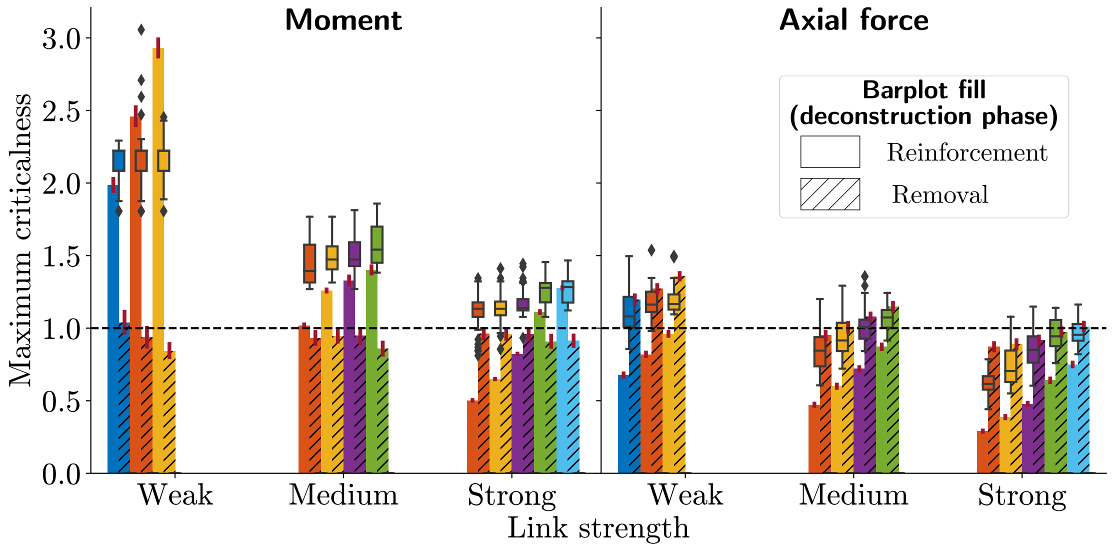


Figure E.12. The effect of varying the agent insertion delay δ on **(a)** the rate of deconstruction, **(b)** the percentage of agents in the structure when the removal phase begins that timeout, and **(c)** the maximum moment and axial force criticalness throughout the deconstruction. The results are shown for 100 trials for bridges of length 2.3 m with strong links. Error bars in **(a)** show the 5th and 95th percentiles. In **(b & c)**, the 95 % confidence intervals are indicated, and in **(c)** the initial cantilever construction and subsequent bridge optimisation are included for comparison.

E.2 Varying Structure Length

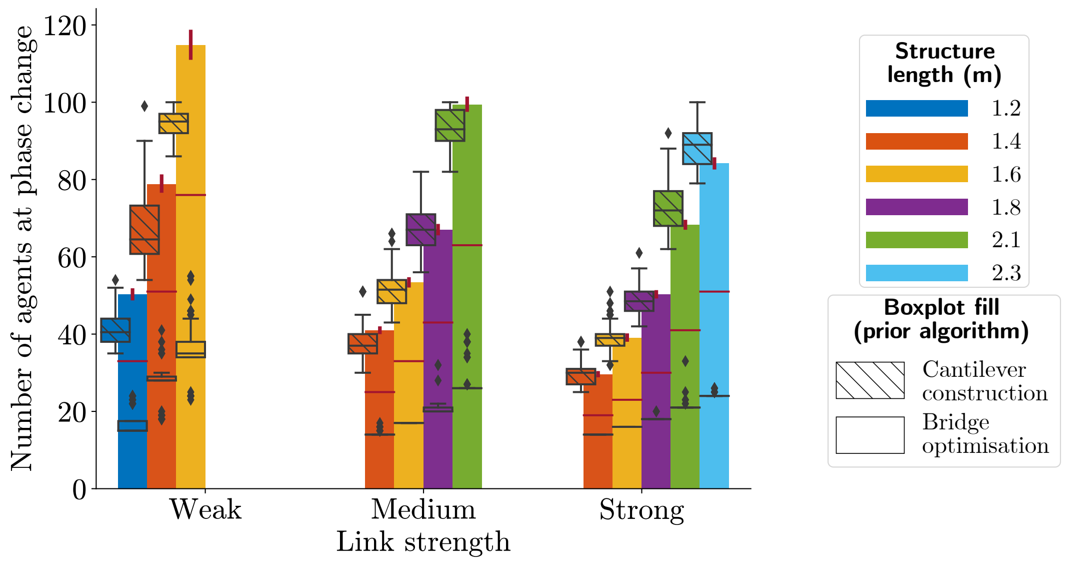


(a)

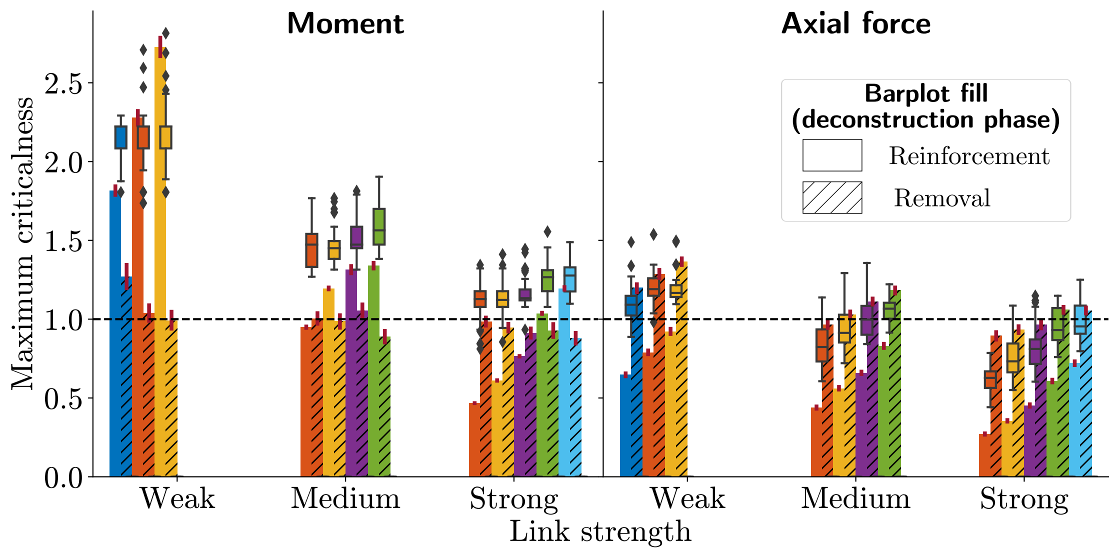


(b)

Figure E.13. The effect of varying structure length on **(a)** the number of agents in the structure when the removal phase begins, and **(b)** the maximum moment and axial force criticalness throughout the deconstruction, showing the differences in the reinforcement and removal phases. Each bar shows the mean of 100 trials for $\delta = 6$ timesteps, and vertical red lines show the 95 % confidence intervals. In **(a)** the optimum number of agents in a stable cantilever of this length is shown by the horizontal red line, and the boxplots show the number of agents in the prior cantilever construction and bridge optimisation stages separately. In **(b)**, the boxplots show the maximum criticalnesses during the prior cantilever construction and bridge optimisation stages combined.

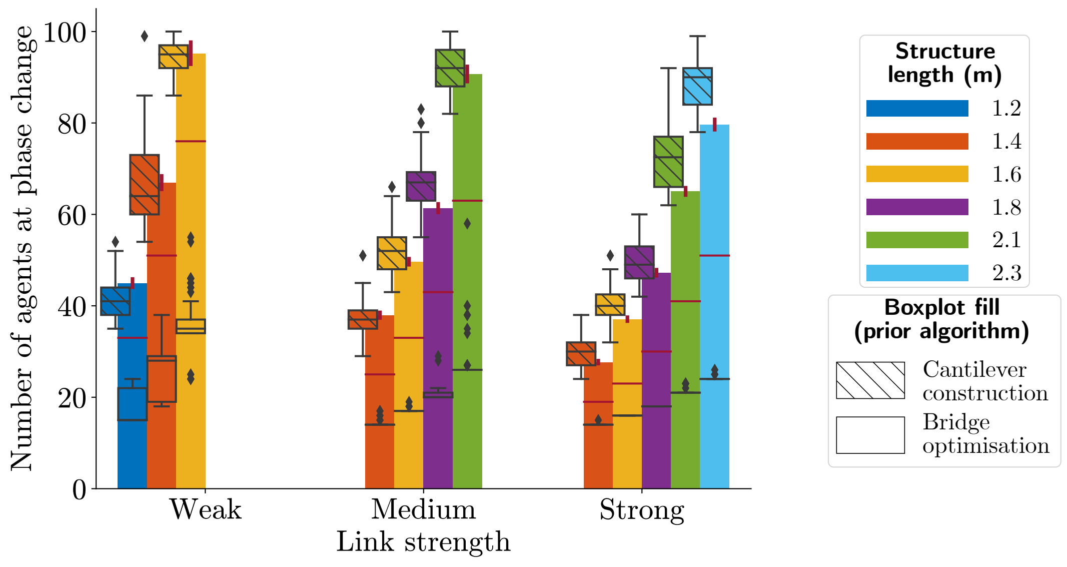


(a)

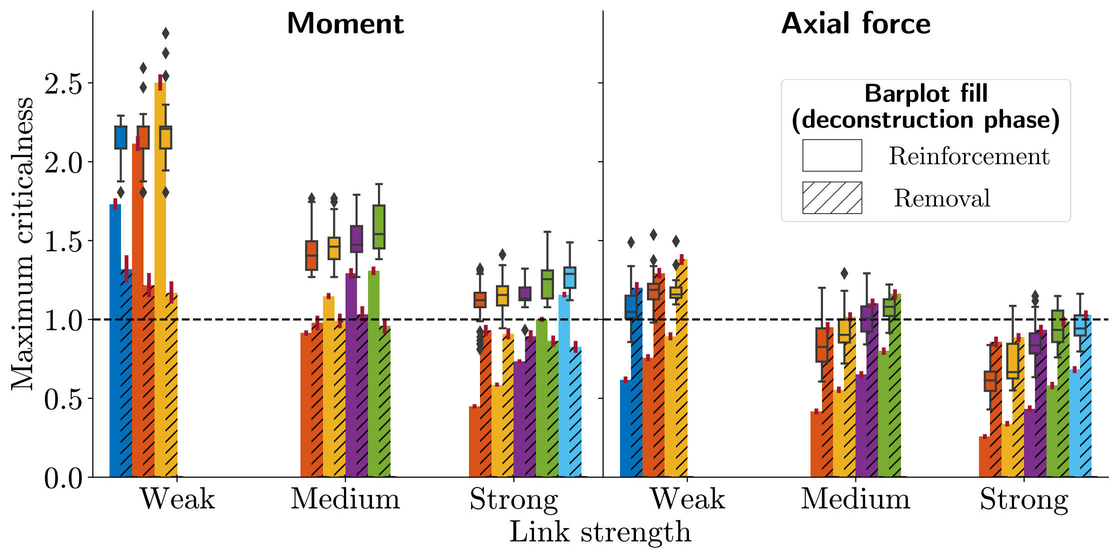


(b)

Figure E.14. The effect of varying structure length on **(a)** the number of agents in the structure when the removal phase begins, and **(b)** the maximum moment and axial force criticalness throughout the deconstruction, showing the differences in the reinforcement and removal phases. Each bar shows the mean of 100 trials for $\delta = 8$ timesteps, and vertical red lines show the 95 % confidence intervals. In **(a)** the optimum number of agents in a stable cantilever of this length is shown by the horizontal red line, and the boxplots show the number of agents in the prior cantilever construction and bridge optimisation stages separately. In **(b)**, the boxplots show the maximum criticalnesses during the prior cantilever construction and bridge optimisation stages combined.

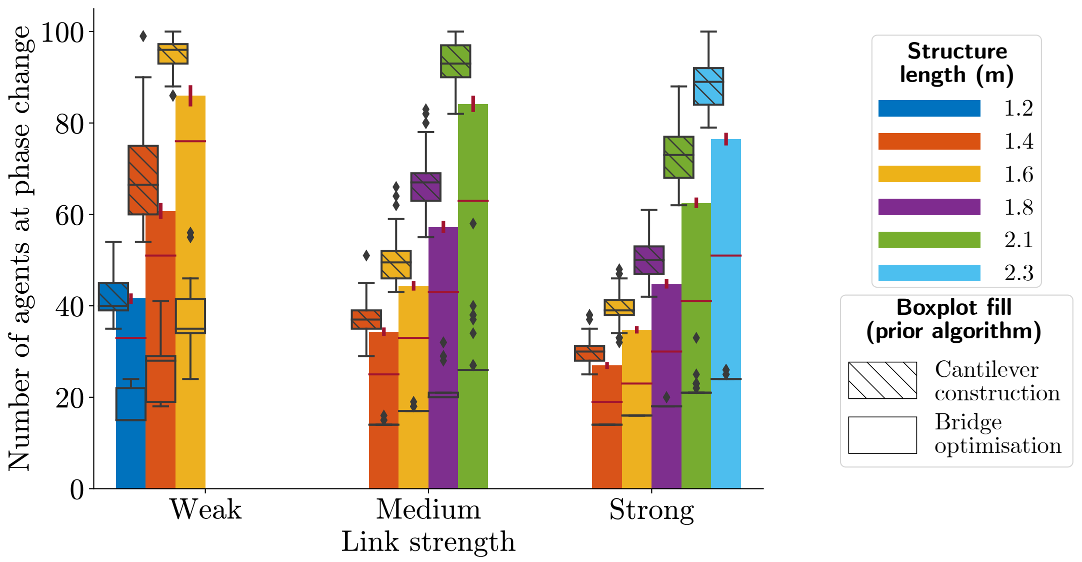


(a)

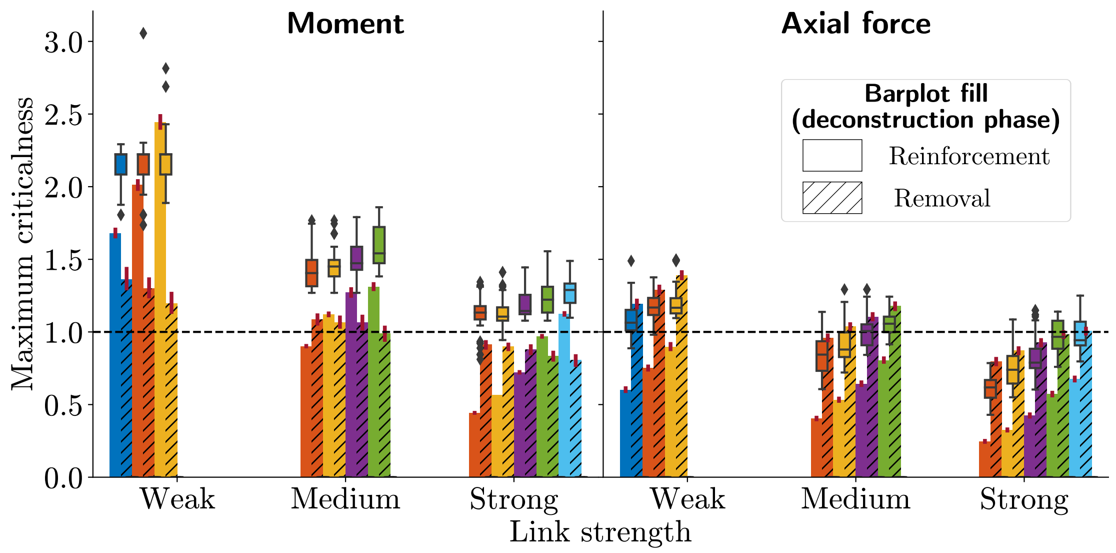


(b)

Figure E.15. The effect of varying structure length on **(a)** the number of agents in the structure when the removal phase begins, and **(b)** the maximum moment and axial force criticalness throughout the deconstruction, showing the differences in the reinforcement and removal phases. Each bar shows the mean of 100 trials for $\delta = 10$ timesteps, and vertical red lines show the 95 % confidence intervals. In **(a)** the optimum number of agents in a stable cantilever of this length is shown by the horizontal red line, and the boxplots show the number of agents in the prior cantilever construction and bridge optimisation stages separately. In **(b)**, the boxplots show the maximum criticalnesses during the prior cantilever construction and bridge optimisation stages combined.



(a)



(b)

Figure E.16. The effect of varying structure length on **(a)** the number of agents in the structure when the removal phase begins, and **(b)** the maximum moment and axial force criticalness throughout the deconstruction, showing the differences in the reinforcement and removal phases. Each bar shows the mean of 100 trials for $\delta = 12$ timesteps, and vertical red lines show the 95 % confidence intervals. In **(a)** the optimum number of agents in a stable cantilever of this length is shown by the horizontal red line, and the boxplots show the number of agents in the prior cantilever construction and bridge optimisation stages separately. In **(b)**, the boxplots show the maximum criticalnesses during the prior cantilever construction and bridge optimisation stages combined.