Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

Distributed Optimisation and Deconstruction of Bridges by Self-Assembling Robots

Edward Bray and Roderich Groß
The University of Sheffield, UK
Email: {enbray1, r.gross}@sheffield.ac.uk

***Abstract*—Multi-robot systems are often made of physically small robots, meaning obstacles that could be overcome by larger robots pose a greater challenge to them. This paper considers how a group of such robots could self-assemble into bridges to cross large gaps in their environment. We build on previous work demonstrating construction of cantilevers to show how they can be modified once the other side of the gap is reached. Two distributed algorithms are presented: one to reduce the number of agents in the initial structure once it is supported at both ends, and another to deconstruct this leaner structure when it is no longer required. A force-aware approach is taken to ensure that structures do not collapse under self-weight. The first algorithm is shown to be capable of reducing the number of agents in the structure to close to the optimum amount, whereas the second achieves safe and reliable deconstruction.**

I. Introduction

Using multiple robots to accomplish a common goal instead of a single robot has benefits in a variety of scenarios, such as those that cover large areas or in which robots could be damaged [8]. Space and cost constraints mean that the robots constituting many of these systems are small, with bodylengths up to tens of centimeters [9, 17, 18, 23]. This presents unique challenges when navigating difficult terrain: they are able to fit through tight constrictions with ease, but struggle to overcome larger obstacles. This paper considers how a group of small robots could work together to cross gaps in the terrain by building bridges. Gaps measuring a few meters across are common in real-world environments, such as woodland or in natural disaster zones, and could be easily passed by large robots, but present a much more significant obstacle to agents with smaller bodies. Bridging these gaps would allow other robots to more easily traverse them. To ensure resources are used efficiently, bridges should consist of minimal building material and be dismantled when no longer required.

This work is inspired by the behaviour of ants, who have been observed to assemble their bodies into different structures for the benefit of the colony with no centralised leader [1], a process called *self-assembly*. Certain species of ants will self-assemble into bridges, either to allow them to reach new locations [7], or to create shorter paths to important locations the colony has already visited [16].

An area of multi-robot system research in which self-assembly is particularly relevant is *modular robotics*, where a finite set of robotic modules come together to form different connected structures to achieve different goals [19]. Previous work has shown how these robots can cross gaps by arranging modules in a line longer than the gap and moving directly over it, as this ensures there are always modules above solid ground [14, 21]. These linear structures are unlikely to be rigid enough to span large distances without sagging, so this approach is only suitable for small gaps.

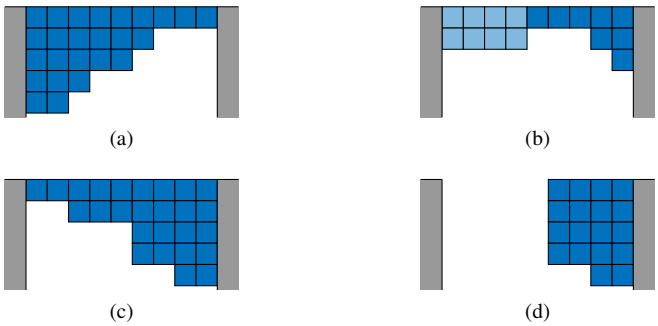

Fig. 1. Bridge optimisation starts with a cantilever that has just reached the other side of a void (a) and agents reconfigure or remove themselves to leave a leaner bridge that will not collapse under its own weight (b). When no longer required, the bridge is deconstructed by first building a structure that would not collapse if detached from the left support (c), then deconstructing this columnwise (d). Agents are shown in blue and fixed supports in grey. The darker portion in (b) shows the equivalent unsupported cantilever, a term introduced in Section IV-B.

Spanning larger gaps requires stronger structures, such as those created by ants. The manner in which robots could self-assemble bridges to reduce travel times to popular locations has been studied in [2, 10]; the latter work also includes prototype soft-bodied robots capable of carrying out this self-assembly. However, these studies do not account for the effect of gravity, so the structures would potentially break in real-life.

For robots to build bridges to access new areas, they must first construct a cantilever that can span across the void without collapsing. One approach is for the robots to build this structure from external building materials, as explored in [11, 12, 13]. These works design custom truss members in parallel with the robots so that they can be effectively manipulated, but these specialist materials must also be transported to the building site. An alternative is to use the robots themselves as the building material, as explored in [3, 20]: this requires a greater number of robots, but the building material can transport itself to the construction site. However, [3] only addresses construction of cantilevers, without considering what should happen when the other side of the void is reached.

Self-assembly of complete bridges is shown in [20], but this approach requires building from both sides of the gap. Neither work considers how the structure can be safely deconstructed.

An intuitive way for a team of robots to build useful structures is for a human to design the desired structure, and tell each agent exactly what they are to build [23]. Another approach is to give high-level goals, such as to span a certain distance without breaking [3, 11, 12, 13, 20]. This can be applied to a wider range of building scenarios than the preplanned approach, and the distributed algorithms increase the scalability of the system while removing the single point of failure of a global controller.

In our previous work [3], we presented distributed algorithms that enable a group of self-assembling robots to build cantilevers that span across a void (Fig. 1), while employing local force measurements to ensure the structure does not collapse under its own self-weight. In this paper, we build on this work by considering two additional problems arising when the cantilever reaches the other side of the void to form a bridge. Firstly, once the structure becomes supported at both ends it is likely to benefit from *bridge optimisation*, where agents are removed or repositioned to produce a structure that is thinner in the middle than at the supports, such that it resembles the arches seen in road bridges (Fig. 1b): this also frees agents to complete other tasks. Secondly, when the structure is no longer required, agents should be able to safely dismantle it, referred to as *deconstruction*. They must first convert the existing bridge to one that is stable when supported on one side (Fig. 1c), before the other side can be released and the structure dismantled (Fig. 1d). This initial step is made more difficult as the guiding force measurements are from a structure supported on both sides, but the agents aim to construct a structure that will be stable when only supported on one side. To the best of the authors' knowledge, neither the optimisation or deconstruction of self-assembled robotic bridges has been investigated in previous studies.

We employ a distributed control strategy that incorporates local force measurements to build structures that do not collapse under their own weight. The forces exerted between interconnected robots could either be measured directly, requiring additional hardware to be incorporated into the links [3, 12, 13], or estimated from knowledge of the structure's configuration, requiring additional computation [15]: the algorithms presented here assume this knowledge is available. The algorithms are implemented in Python using the same custom simulator used in [3] that approximates the structure as a truss to efficiently calculate the internal forces [22].

II. Problem Formulation

This work considers a 2D grid of homogeneous agents of side length l that can connect to each other through *links* on any face: links along rows and columns are called *row* and *column links* respectively. Agents are initially arranged in a configuration of length L between two *fixed support* surfaces, to which agents can also connect (Fig. 1a). Locations in the grid are referenced as $(row, column)$, where the agents in the top left and top right of the structure occupy locations $(1, 1)$ and $(1, L)$ respectively, and row number increases downwards.

Agents can either be $placed$ or $active$. Placed agents do not move, but can become active when they deem it necessary. Structures must be *continuous*, meaning that each placed agent connects 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. While connected to both supports, it must also contain a single contiguous region with only one placed agent in each column, referred to as the *narrow section* (columns 5–7 in Fig. 1b). Note that 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 6 in Fig. 1b). These restrictions are imposed so that structures resemble arches, a strong shape commonly found in bridges. We term empty locations with a placed agent on either side *canyons*. Active agents cannot place in canyons without violating the requirement for a narrow section.

In each simulation *timestep*, active agents can travel along the perimeter of the structure by moving into an unoccupied cell in their Moore neighbourhood if in doing so they will remain adjacent to another agent. Active agents are required to be able to pass between the top and bottom perimeters of the structure, which could be achieved by either moving into the third dimension in front of existing placed agents, or by coordinating placed agents to shift down the column in question by one cell. We choose the former method as it requires less coordination, but do not allow moving into the third dimension in other situations as constructing 3D structures is beyond the scope of this work.

Each agent has self-weight w acting downwards, and is able to measure the moment M and axial force F in its links. *Allowable limits* $M_{allowable}$ and $F_{allowable}$ are set below the failure strength of the links, from which the *criticalness* γ of each link is calculated as $\max\left(\gamma^M, \gamma^F\right)$ [3] where:

$$\gamma^M = \frac{|M|}{M_{allowable}} \qquad \gamma^M = \frac{\max(F, 0)}{F_{allowable}}$$

This formulation represents links that are equally strong to bending in all directions, strong in compression, and weak in tension. We assume that the shear strength is high in comparison, so can be neglected. This is similar to a simple connection mechanism such as that of SMORES [5].

When $\gamma \geq 1$, links are described as *critical*, indicating that they may be close to failure. Structures in which there are no critical links are deemed *stable*, otherwise they are *unstable*. This work considers how structures can be reconfigured and deconstructed, while aiming to maintain stability. Specifically, the bridge optimisation algorithm aims to reduce the number of agents in an initial configuration spanning between the two fixed support surfaces, and the deconstruction algorithm deconstructs the structure and recovers agents on the opposite side of the void to where the initial bridge was build from. We set $l = 0.1\,\mathrm{m}$ and conservatively assume robots are a cube of solid aluminium, giving $w = 19.3\,\mathrm{N}$: our agents are therefore similar in size to existing robotic platforms, but heavier [5].

III. Offline Structural Optimisation

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

Optimal configurations are generated by exhaustive search, which begins by selecting an L, N, $M_{allowable}$, and $F_{allowable}$. 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 right supports N_L and N_R respectively. We iterate over $\lceil \frac{N-L}{2} \rceil \leq N_L \leq (N-L)$ and enumerate all configurations of N_L agents using integer partitioning [6], where the i^{th} component of the partition represents the number of agents in row $i+1$ connected to this 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 also enumerated, giving the number of agents in row $i+1$ connected to the right support $N_{R,i}$. We arrange the partitions such that $N_{L,i} \geq N_{L,i+1}$ and $N_{R,i} \geq N_{R,i+1}$ to satisfy the continuity condition, and discount non-physical configurations in which $N_{L,i}+N_{R,i} > L$. It is possible that $N_{L,i}+N_{R,i} = L$, thus we do not require a narrow section 1 agent high.

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, we set $L = N = 1$ to find the minimum N required to build stable structures of $L \geq 1$.

IV. Algorithm Design

The following sections describe the algorithms that achieve bridge optimisation and deconstruction. Section IV-A gives an overview of the bridge optimisation algorithm, and Section IV-B presents the changes made to it in the deconstruction algorithm. Section IV-C describes how elastic beam theory is used to improve the performance of the deconstruction algorithm. Sections IV-D and IV-E describe aspects of the algorithms in more detail.

Active agents *advance* each timestep in a randomised order, as in the parallel cantilever construction algorithm of [3]. They 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. M and F in links is updated for all agents at the end of each timestep.

A. Bridge Optimisation

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 structure's strength. Every timestep, each eligible agent j will release itself from the structure with probability $P_{release}$. This is calculated from the maximum γ across each link of agent j, $\gamma_{j,max}$, as:

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

This function has two shape parameters: α affects the rate the function decays with $\gamma_{j,max}$, and $\beta = P_{release}(0)$. Throughout this work, we set $\alpha = 10$ but explore the effect of changing β to examine how the number of simultaneous active agents in the simulation affects performance. Each agent that releases itself becomes active and disconnects all but one of its links so that it is only connected to the structure by a single link thus isn't offering any support (see Section IV-D). In the next timestep, they begin to follow Algorithm 1.

Active agents start in the `releasing` mode, where they first check the readings of M and F in links of the agent above. If their release has caused any of these links to become critical, they immediately place back here. Otherwise they swap to the `gathering` mode and make their first step (lines 2 – 9).

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). Agents travel left until they reach column 1, then move right to column L to obtain information about the whole structure (line 20). If no links were measured to be critical when they reach column L, they will swap to the `escaping` mode (line 14), in which they pass through the structure here to reach row 0, then move to the right support, where they exit the simulation (lines 21 – 25). If an agent believes any links are critical, it will attempt to reinforce them. This begins with calculating a probability mass function $p_{col}(c)$ as described in [3] that represents the probability of placing in column c based on the received force information: 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 [3]. It 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 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:

(i) The agent is not in a canyon and placing here will not violate the continuity condition, so it does so (line 30).

(ii) The placement location is at the top of a canyon, so the agent swaps to the `escaping` mode and will pass through the structure here instead (line 33). This decreases the total number of active agents quickly when there is too much traffic to make informed decisions about placement locations, and reduces congestion around canyons.

(iii) Placing in this column violates the continuity condition, so the agent will select another c_{target} from $p_{col}(c)$ without replacement (line 35).

Algorithm 1: The procedure followed by active agents during bridge optimisation (based on [3], Algorithm 2)

```
1  switch mode do
2    case releasing do
3      Read M and F from placed agent above;
4      if Agent 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 M and F from placed agent above;
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_col(c) from recorded M & F;
17         c_target ← sample from p_col;
18         mode ← placing;
19     else
20       Make step;
21   case escaping do
22     if In position (0, L) then
23       Agent leaves simulation;
24       return
25     Make step;
26   case placing do
27     if In column c_target then
28       Attempt to place;
29       if Placement succeeded then
30         Agent no longer active;
31         return
32       else if Agent at top of canyon then
33         mode ← escaping;
34       else
35         c_target ← sample from p_col;
36     Make step;
37   case swapping do
38     mode ← previous mode;
39 if Agent stationary for > τ timesteps then
40   Attempt placement;
41   if Placement succeeded then
42     Agent no longer active;
```

If the agent is subsequently still active, it will move towards the top of 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 [3]. 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). Section IV-E explains this procedure in more detail.

If an agent is stationary for too long, it is deemed to have got stuck and attempts to place itself where it is (lines 39 – 42). Here, the timeout τ is set to 20 timesteps.

B. Deconstruction

In the deconstruction algorithm (Algorithm 2), agents transition through broadly the same modes as in bridge optimisation, but with the changes and additions described below; line numbers in this section refer to Algorithm 2. We assume the bridge was built from the left side of the void, so agents finish on the right side after deconstruction. There are two phases:

(i) *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 Fig. 1b to 1c). The narrow section and the columns to its right are referred to as the *equivalent unsupported cantilever* (the darker region in Fig. 1b).

(ii) *Removal*: The structure disconnects from the left support and agents leave it above the right support (Fig. 1d).

As we will see in Section IV-C, agents must be able to measure the shear force S in their links in addition to M and F.

An additional mode, `force-releasing`, 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 deconstruct canyons as explained below. When such an agent releases, they either enter the `gathering` or `escaping` mode, depending on the reason they were instructed to release: 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 Figs. 2a and 2b). 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 release themselves with probability $P_{release}$ as in the bridge optimisation algorithm, with $\beta = 0.2$. Agents on the right of the inflection point cannot do this, as we assume that they are placed in a satisfactory position. Secondly, additional `gathering` active agents enter the simulation in location $(0, L)$ a fixed number of timesteps δ apart, assuming this location is not already occupied, until the first `escaping` agent occupies this cell. They then travel along the top of the structure, obtaining force information from the agent below them as they move (line 16), 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. `gathering` agents on the lower perimeter head directly for column L without first visiting column 0 (line 17).

When a `gathering` agent on the lower perimeter reaches column L, it calculates $p_{col}(c)$ (line 18) and switches to the `placing` mode (line 23) regardless of the M and F measurements it received. If it believes the structure is unstable, it draws c_{target} from $p_{col}(c)$ as before (line 22). However, it no longer transitions to the `escaping` mode if it believes the

Algorithm 2: The bridge deconstruction algorithm, emphasising differences with bridge optimisation.

```
1  switch mode do
2    case force-releasing do
3      if Able to release then
4        mode ← escaping;
5      else if Continuity condition would be violated by
         release then
6        Set agent below to force-releasing;
7    case releasing do
8      Read M and F from placed agent above;
9      if Agent above has a critical link then
10       Place back here (agent no longer active);
11       return
12     else
13       mode ← gathering;
14       Make step;
15   case gathering do
16     Record M and F from placed agent above or
         below;
17     if Agent in column L then
18       Calculate p_col(c) from recorded M & F;
19       if No measured links critical then
20         c_target ← leftmost column in narrow section;
21       else
22         c_target ← sample from p_col;
23       mode ← placing;
24     else
25       Make step;
26   case escaping do
27     if In position (0, L) then
28       Agent leaves simulation;
29       return
30     Attempt to release agent below if required;
31     Make step;
32   case placing do
33     if In column c_target then
34       Attempt to place or release adjacent agent;
35       if Placement succeeded then
36         Agent no longer active;
37         return
38       else if Released tip then
39         mode ← escaping;
40       else
41         c_target ← sample from p_col;
42     Make step;
43   case swapping do
44     mode ← previous mode;
45 if Finished in canyon then
46   Set agent on left to force-releasing to gather;
47 if Agent above force-releasing to escape then
48   c_target ← 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 ← escaping;
53 if Agent stationary for > δ steps then
54   Attempt placement;
55   if Placement succeeded then
56     mode ← placed;
```

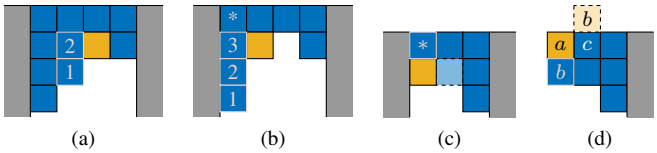

Fig. 2. Forcing agents to release. Placed agents are shown in blue, active agents in yellow, the fixed supports in grey, and force-releasing agents have a cream border. (a) The active agent is in a canyon, so the column to the left should be released. (b) The active agent believes the unsupported cantilever is stable, so can release another column left of the narrow section. Alternatively, if the agent marked with an asterisk is force-releasing to escape, the agents below will release first. In both (a) and (b), agents are blocked by others below them, so they release in the numbered order shown. (c) The agent marked by an asterisk is released and deconstruction begins: the pale blue cell is either empty or a placed agent. (d) Agent a is escaping, so sets agent b to force-releasing when it moves off. When agent b reaches the position shown paler, it will set agent c to force-releasing.

structure is stable. It instead 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 this column in the placing mode, it sets the agent on its left to force-releasing to gather if it is still the first one left of the narrow section, instead of attempting to place (line 34). If $c_{target} = 1$ here, the active agent instead sets the agent above to force-releasing to escape, thus initiating the removal phase (Fig. 2c with the pale blue cell unoccupied) and enters the escaping mode itself (lines 38 – 39). Agents draw from $p_{col}(c)$ (line 41) in two scenarios: if the placement was not possible due to violations of the continuity condition, as in the bridge optimisation algorithm, but also after attempting to release an adjacent agent. In this algorithm $p_{col}(c)$ is calculated from measurements in links along the top and bottom of the structure, and is set to 0 left of the narrow section.

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 at the bottom of column c reaches the top of column $c+1$, it triggers the agent at the top of this column to release (line 30 and Fig. 2d). If there are agents below one force-releasing to escape that are not connected to the right side of the structure, then these agents will become force-releasing to gather first to maintain continuity (Fig. 2b if the asterisked agent is force-releasing to escape).

Agents still timeout if they have been stationary for too many timesteps, but now $\tau = \delta$ (line 53). Before assessing this, each agent checks those around it to see if any additional behaviours should be triggered. If the agent is inside a canyon, it will tell the one to its left to enter the force-releasing mode to gather (lines 45 – 46 and Fig. 2a), so that the equivalent unsupported cantilever is extended and congestion

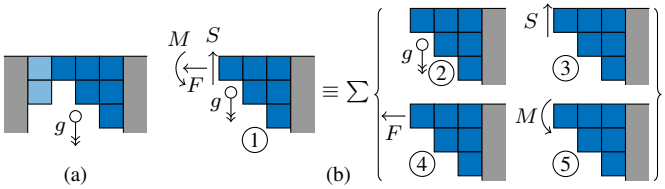

Fig. 3. (a) Cutting off the columns to the left of the narrow section (shown paler) reveals the internal shear force S, axial force F, and moment M. This represents case ① of (b), which can be decomposed into the sum of the separate loading cases ② – ⑤. The double-headed arrow g denotes the acceleration due to gravity, and the fixed supports are shown in grey.

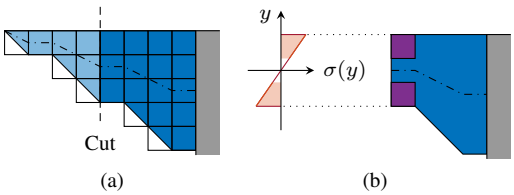

Fig. 4. Approximating the stress force distribution due to an arbitrary known loading (not shown), with the fixed support shown in grey. (a) The outlines of a collection of agents arranged as a cantilever, where the filled region denotes the cantilever profile used to calculate the approximate force 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.

around canyons is reduced. If the agent is below one that is `force-releasing` to escape, then 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): it therefore leaves the area below where the removal phase is occurring without obstructing other agents. If the agent is in position $(2, 1)$ and there is a placed agent to its right, it initiates the removal phase (lines 49 – 50 and Fig. 2c with the pale blue cell occupied) as we assume the structure is unlikely to get any more stable. Finally, if the agent ends its step in row 1 or left of one that is `force-releasing` to escape, it swaps to `escaping` (lines 51 – 52).

C. 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, so they should place themselves so as to reduce the forces in the equivalent unsupported cantilever. This requires each agent to convert the forces in the bridge into what they would be in this cantilever. It would be possible to calculate this using the truss approach employed by 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 by each agent using the principle of superposition [4]. The bridge in Fig. 3a can be split on the left face of column 2 to give the equivalent unsupported cantilever (shown darker), which has the loads shown in case ① of Fig. 3b. Agents measure M and F under this loading configuration, but would like to obtain M and F for the equivalent unsupported cantilever under gravity, loading case ②. This can be approximated by subtracting the force distributions under loading cases ③ – ⑤ from case ①.

The active agent is informed of measurements of S, F, and M made by the placed agent at this location. Therefore the stress distribution under these additional loading configurations can be approximated by elastic beam theory [4] once the active agent reaches column L as it has 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 (Fig. 4a), which is analysed using elastic beam theory to calculate the distribution of the longitudinal stress σ across the faces of each column of agents. Equivalent M and F in the links of the top and bottom agents in each face are calculated from this distribution (Fig. 4b). For these loads, σ is a polynomial function of the structure's geometric parameters and the known loads S, F, and M, so this calculation is quick to make. Note that forces can only be superimposed like this for linear elastic materials deflecting a small amount [4], and we are only modelling a simplified case of the stress on this face 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 $p_{col}(c)$ so that agents place in columns more suitable to reinforce the equivalent unsupported cantilever.

D. Active Links

In order to prevent active agents providing support to the structure, they choose a single face to attach with by their *active link*. If an agent is connected to another agent's active link 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 normal connection face, or connecting on this face would prevent movement in the next step, as described below.

1) 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.

2) Above or in row 1: The active link of agents at the unsupported tip in row 1 is always set to the right face (Fig. 5a, purple). The active link of agents in row 0 will be the right face if no agent is below it (Figs. 5a & 5b, green).

3) Canyons: When the active agent is inside 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

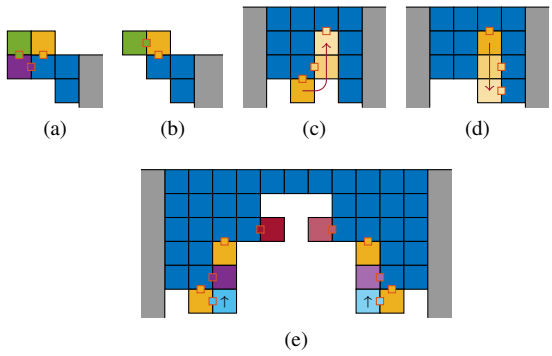

Fig. 5. Setting active links in special circumstances. Placed agents are shown in blue, fixed supports in grey, and active agents in other colours. Each agent's active link is shown by the small orange-bordered square of matching colour. (a) and (b) show scenarios when active agents are around the tip. (c) and (d) show an active agent entering and leaving a canyon, where the colour gets paler to denote the position in subsequent timesteps as the red motion arrow is followed. (e) shows the situations considered in cantilever construction, and their equivalents shown in paler colours when the same situation occurs on the other side of the inflection point (adapted from [3], Fig. 4).

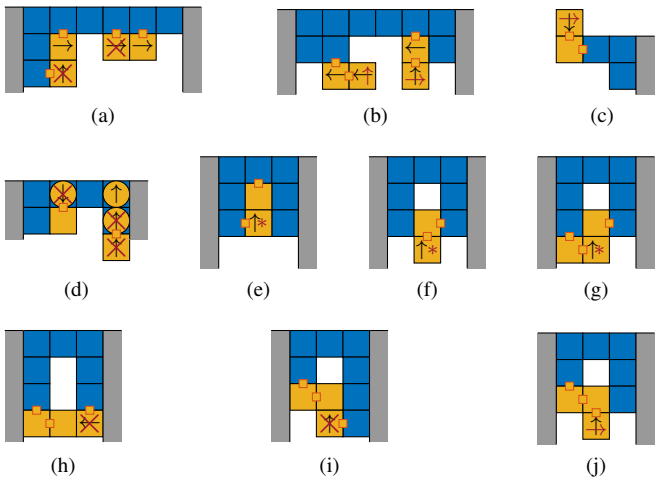

Fig. 6. Special behaviours when agents should not swap information with each other. Active agents are shown in yellow (as a circle if they are passing through the structure), placed agents in blue, and the fixed supports in grey. Active links are shown as small yellow squares with orange borders. Salient directions of travel are shown by black arrows, and overrides are shown in red. Agents marked with a red asterisk will skip over the canyon instead of entering it. Equivalent behaviours exist for the same situations mirrored about the vertical axis.

they reach the top of the canyon. The active link is set to the top face when at the top of the canyon, and otherwise to the left face when on the left side of the inflection point and vice versa (Figs. 5c & 5d).

4) Mirroring of cantilever cases: During cantilever construction, the active link is set to the left face if either (i) there is no agent above, (ii) there is an active agent above and a placed agent on the left, or (iii) the agent in question is moving upwards, there is an active agent to its left, and an active agent above it that is not swapping [3]. Here we include the mirror of these conditions about the vertical axis to allow agents to set their active link to their right face when right of the inflection point (Fig. 5e red, purple, and cyan respectively).

E. 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 each other to move into, as outlined below.

1) Direction: Since agents advance in a random order, it is possible that two agent 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 (Fig. 6a) 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 (Fig. 6b).

2) Around the tip: As explained in Section IV-B, any agents at the tip during the deconstruction algorithm will swap to the `escaping` mode. If an agent attempts to swap with an `escaping` agent, it will also switch to the `escaping` mode, thus its direction will reverse as in Fig. 6c.

3) 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 (Fig. 6d). Each algorithm only allows agents to pass through the structure in one direction, so this is a special case of (1).

4) Within canyons: Canyons represent a significant bottleneck in agent motions as each would like to pass up and down the whole canyon. The total time that colliding agents spend in canyons is reduced by not swapping agents here, and instead making the lower agent pass over the canyon without visiting the top unless strictly necessary. The agents marked by an asterisk in Figs. 6e – 6g are currently on the left of the canyon, and would usually travel up it, but instead swap to the right of the canyon and carry on with their motion. If the lower agent is in the `gathering` mode, the higher agent will 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 $p_{col}(c)$ in the bridge optimisation algorithm. In the deconstruction algorithm, the swap can continue if the lower agent will set the agent on the left of this column to `force-releasing`, otherwise it scales $p_{col}(c)$ to be 0 in and left of the canyon, then draws another c_{target}.

5) Different sides of canyon: It is also possible that 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 (Figs. 6h & 6i). If the agents are attached, the lower one will take a step backwards to allow the other to vacate the space in the next timestep (Fig. 6j).

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

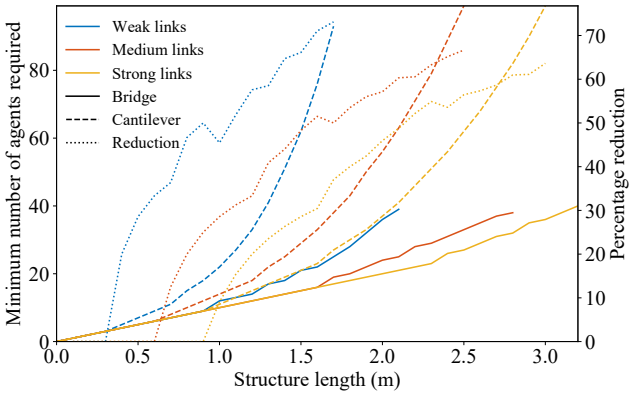

Fig. 7. The minimum number of agents required to build stable bridges (solid lines) or cantilevers (dashed lines) of a given length. Dotted lines show the percentage of agents in the cantilever that can be removed to leave an optimal bridge of the same length.

V. Simulation Results and Discussion

The algorithms were tested for the same link strengths as the cantilever construction in [3]: *weak*, *medium*, and *strong* links refer to $[M_{allowable}, F_{allowable}] = [13.9\,\mathrm{N\,m}, 579\,\mathrm{N}]$, $[42.6\,\mathrm{N\,m}, 1159\,\mathrm{N}]$, and $[86.9\,\mathrm{N\,m}, 1738\,\mathrm{N}]$ respectively. Weak links were tested for initial configurations with $L \in \{1.2, 1.4, 1.6\}$m, links of medium strength were tested for $L \in \{1.4, 1.6, 1.8, 2.1\}$m, and strong links were tested for $L \in \{1.4, 1.6, 1.8, 2.1, 2.3\}$m. The number of active agents in each simulation at a time was varied in bridge optimisation by setting $\beta \in \{0.01, 0.05, 0.1, 0.2, 0.3\}$, and in deconstruction by setting $\delta \in \{6, 8, 10, 12\}$. 100 trials were performed for each setting, with simulations stopping when either 50 steps passed without the structure changing, or no agents remained.

A. Offline Optimal Bridges

Optimal bridges were generated offline for each link strength for $N \leq 40$ agents. As shown in Fig. 7, this allows for optimal bridges to be compared to all lengths of optimal cantilever generated in [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, with the difference increasing for longer structures. For the structures tested here, up to 73% of the agents in an optimal cantilever can be removed to create an optimal bridge of equal L. For all optimal bridges found, $N < 2L$ so only the first row can span the whole gap: this implies that additional full rows are inefficient for the L considered here, supporting the constraint that all valid bridges produced by the algorithms include a narrow section.

B. Forces in the Equivalent Unsupported Cantilever

An example of calculating the forces in the equivalent unsupported cantilever through superposition is shown in Figure 8, in which values of M and F in row links along the top and bottom of the structure are plotted with and without the superposition correction applied to the measurements. It is seen that the correction is not perfect, but the difference with the unsupported cantilever case 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. 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 so it is harder for agents to differentiate between columns that it would be effective to place in or not.

C. Bridge Optimisation

Each trial of the bridge optimisation algorithm was initialised from a different randomly-selected trial of the parallel cantilever self-assembly algorithm of [3] with the same link strength and 10 timesteps between agents when the structure reaches length L. The Supplementary Material contains animations of example bridge optimisation trials. They show that trials typically begin with a large number of agents releasing, as the structure is initially configured inefficiently. The link connecting the structure to the right support is often critical to begin with, but is quickly reinforced. The remaining active agents leave the structure, occasionally forming queues, particularly around canyons, but the algorithms are observed to avoid deadlocks in all trials. Over time, the number of active agents decreases and the changes in the structure become more minor. The final configuration contains significantly fewer agents than the initial configuration. The Supplementary Material also shows a trial where agents travel along the top of the bridge as it is reconfigured, illustrating how the bridge enables other agents to explore new areas, and showing the robustness of the algorithm to different scenarios.

The average performance of the bridge optimisation algorithm is shown in Fig. 9. 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 (Fig. 9a). This Figure is drawn for links of medium strength and $L = 1.6\,\mathrm{m}$, but the behaviour is similar for all tested parameter settings. Larger probabilities of leaving result in faster optimisation, but the effect is minimal for $P_{release} > 0.2$. The cost for this faster optimisation is higher maximum γ during construction. This occurs as there are more agents moving around the structure without providing any support to it at any time, thus the measured M and F values are typically higher than if there are only a few active agents moving at once. This means the force information the active agents receive is outdated by the time they reach a position to reinforce the structure. Additionally, the agents sometimes cannot reach the best locations before timing-out due to congestion.

Figs. 9b and 9c show results across different L for $P_{release} = 0.2$. Fig. 9b shows that the number of agents remaining after optimisation is significantly lower than the initial configuration for all parameter settings. The final number

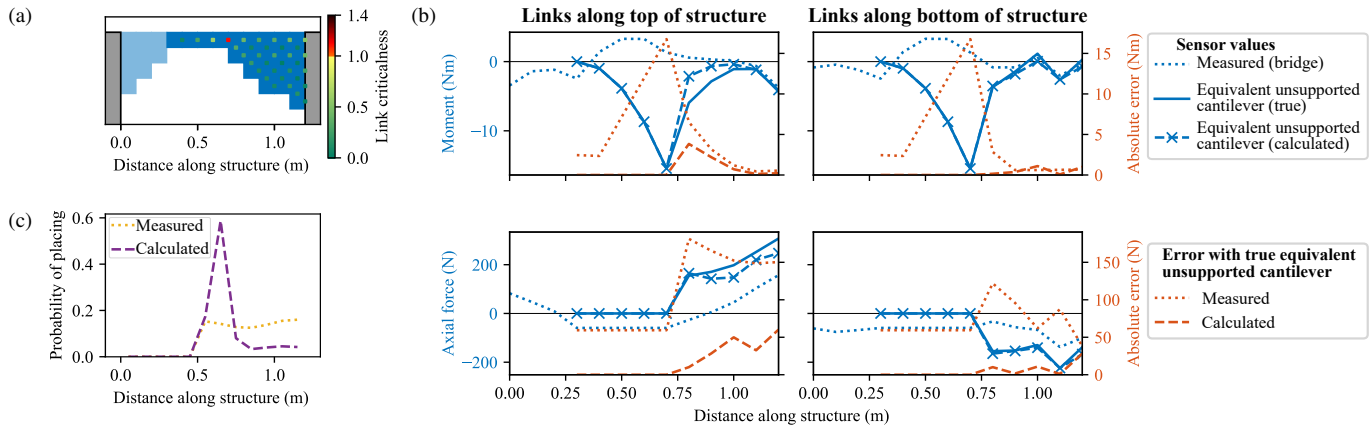

Fig. 8. (a) An example bridge highlighting its equivalent unsupported cantilever (darker): links are weak and coloured by criticalness. (b) The measured values of M and F in row links along the top and bottom of this structure are compared to the true values that would be measured in the equivalent unsupported cantilever, and the values calculated by superposition. (c) The probability of placing in each column as calculated from the measured values and the values calculated by superposition.

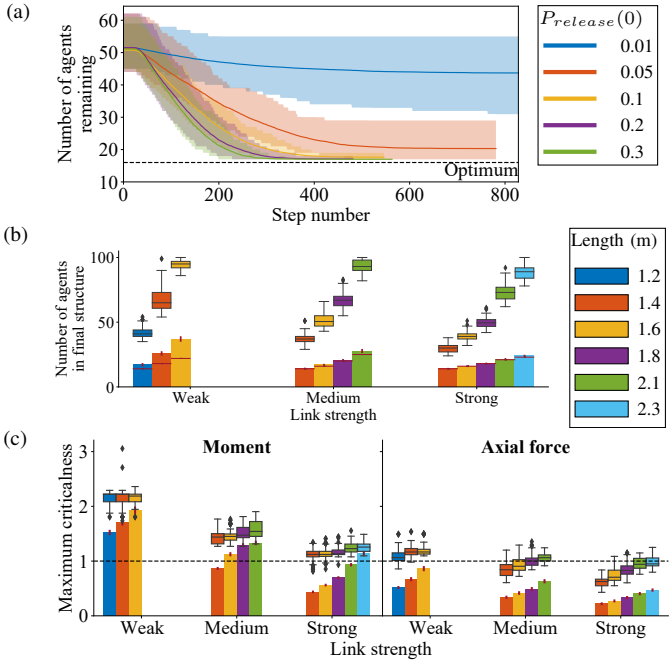

Fig. 9. Performance of the bridge optimisation algorithm shown for different bridge lengths. Results are averaged over 100 trials. (a) The effect of varying $P_{release}(0)$ (given the symbol β) on the rate of agent removal, shown for a bridge of length 1.6 m with links of medium strength; error bars show the 5$^{\text{th}}$ and 95$^{\text{th}}$ percentiles. (b) and (c) show trials with $\beta = 0.2$ with vertical red lines show the 95% confidence intervals, and boxplots showing the equivalent data during construction of the cantilevers used as the starting configurations. (b) The number of agents remaining in the bridge after optimisation; horizontal red lines show the optimum number of agents. (c) The maximum moment and axial force criticalness throughout the optimisation.

of agents is closer to the optimum for stronger and shorter structures. Fig. 9c shows that the maximum γ during bridge optimisation is typically lower than during the initial cantilever construction, so assuming the cantilever construction is safe, the bridge optimisation also will be.

D. Deconstruction

Trials of the deconstruction algorithm were initialised from a different random trial of the bridge optimisation algorithm for $\beta = 0.2$. Animations of the deconstruction algorithm are presented in the Supplementary Material. They show how active agents initialise and 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 typically observed just before and after the structure is released from the left support, but it is quickly reduced as the algorithm enters the removal phase. There is then a constant flow of agents exiting the simulation at a high rate.

Fig. 10 shows the average behaviour of the algorithm. The effect of varying δ is shown for links of medium strength with $L = 2.1\,\text{m}$ in Fig. 10a. This 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 around the structure, meaning more agents are required before the removal phase can begin. However, the phase change occurs earlier in the simulation, so since agents leave the structure in the removal phase at the same rate regardless of δ, the overall number of steps taken to deconstruct the structure is similar for all δ tested. More agents were required during the reinforcement phase than in the optimal cantilever of this length, but the discrepancy decreases with higher δ. Similar behaviour is observed for the other L and link strengths tested.

Comparisons are also made across different L and link strengths by holding $\delta = 10$ timesteps. Fig. 10b shows the number of agents in the structure when the removal phase begins under these circumstances, illustrating how structures that are longer or have weaker links require more agents. The number of agents required is greater than in both the optimal

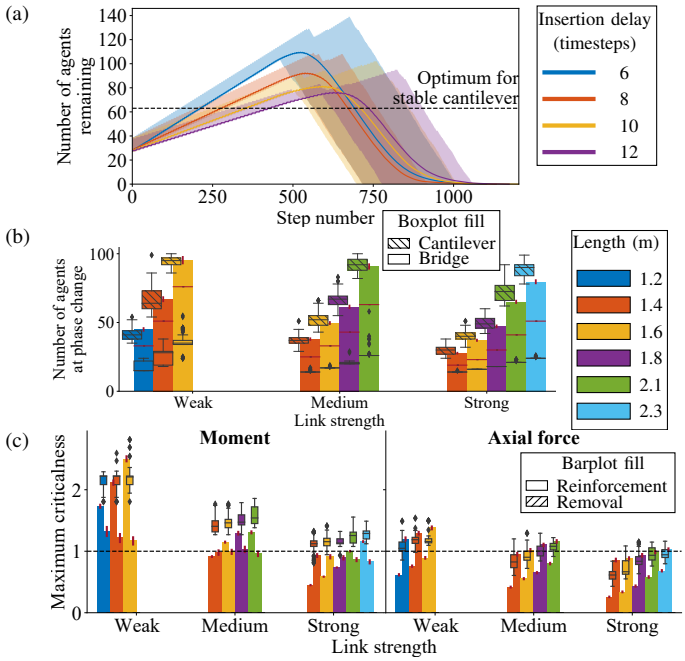

Fig. 10. Performance of the deconstruction algorithm shown for different bridge lengths. Results are averaged over 100 trials. (a) The effect of varying the delay between agents being inserted δ on the rate of deconstruction, shown for a bridge of length 2.1 m with links of medium strength; error bars show the 5$^{\text{th}}$ and 95$^{\text{th}}$ percentiles. (b) and (c) show trials with $\delta = 10$ timesteps with vertical red lines showing the 95% confidence intervals. (b) The number of agents in the bridge when the removal phase begins; horizontal red lines show the optimum number of agents in a stable cantilever of this length, and boxplots show the agents in the prior cantilever construction and bridge optimisation stages separately. (c) The maximum moment and axial force criticalness throughout the deconstruction, showing the differences in the reinforcement and removal phases; boxplots show the maximum criticalnesses during the prior cantilever construction and bridge optimisation stages combined.

cantilever and the original bridge, but similar to that in the original cantilever.

Fig. 10c 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 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 these locations are not necessarily good for reducing γ^M at that instant. In contrast, the maximum γ^F occurs during the removal phase, indicating that agents do not place themselves to efficiently reduce γ^F in the equivalent unsupported cantilever: the animations in the Supplementary Material show this high γ^F normally occurs in the top right corner of the structure. These behaviours can be explained as the largest errors in the approximated M and F are seen for F in links along the top of the structure close to the right support (Fig. 8). These errors means that agents do not place in locations that effectively reduce γ^F here.

VI. Conclusion

This paper presents two distributed algorithms to reconfigure self-assembled structures that bridge a void, while taking into account local force information to build structures that will not collapse under self-weight. The first optimises the initial bridge by reducing the number of agents within it to allow these agents to accomplish other tasks. The second enables the structure to be safely deconstructed when no longer required, leaving all agents on the opposite side of the void to where they began.

The bridge optimisation algorithm is shown to remove the majority of the agents from the original configuration to leave a structure that is 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 during construction of the initial bridge. The deconstruction algorithm is capable of successfully deconstructing all the structures tested, usually requiring a similar number of agents to the non-optimised bridge, but often exceeding the maximum criticalness experienced during its construction. Having fewer simultaneous active agents led to the bridge optimisation algorithm creating bridges with fewer agents, and to the deconstruction algorithm requiring fewer agents to safely disconnect from the left support. Longer structures and those with weaker links required more agents and experienced greater criticalness during the operation of each algorithm.

Future work could account for the dynamic forces incurred as agents move from one position to the next. The algorithms could also be extended to 3D to explore a wider and more realistic range of structure configurations. Another area for future work would be to implement these algorithms on real robotic hardware to explore how they perform in the real-world.

References

[1] Carl Anderson, Guy Theraulaz, and J-L Deneubourg. Self-assemblages in insect societies. *Insectes sociaux*, 49(2):99–110, 2002.

[2] Marta Andrés Arroyo, Sarah Cannon, Joshua J Daymude, Dana Randall, and Andréa W Richa. A stochastic approach to shortcut bridging in programmable matter. *Natural Computing*, 17(4):723–741, 2018. URL https://link.springer.com/article/10.1007/s11047-018-9714-x.

[3] Edward Bray and Roderich Groß. Distributed Self-Assembly of Cantilevers by Force-Aware Robots. In *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 110–118. IEEE, 2021. URL https://ieeexplore.ieee.org/abstract/document/9620697.

[4] Stephen H Crandall, Norman C Dahl, and Thomas J Lardner. *An Introduction to the Mechanics of Solids*. McGraw-Hill, 2nd edition, 1978. ISBN 0-07-013441-3.

[5] Jay Davey, Ngai Kwok, and Mark Yim. Emulating self-reconfigurable robots-design of the SMORES system. In *2012 IEEE/RSJ international conference on intelligent*

robots and systems, pages 4464–4469. IEEE, 2012. URL https://ieeexplore.ieee.org/abstract/document/6385845.

[6] David Guichard. An introduction to combinatorics and graph theory. *Whitman College-Creative Commons*, 2020.

[7] Bert Hölldobler and Edward O Wilson. The multiple recruitment systems of the African weaver ant *Oecophylla longinoda* (Latreille)(Hymenoptera: Formicidae). *Behavioral Ecology and Sociobiology*, 3(1):19–60, 1978. URL https://link.springer.com/article/10.1007/BF00300045.

[8] Belkacem Khaldi and Foudil Cherif. An overview of swarm robotics: Swarm intelligence applied to multi-robotics. *International Journal of Computer Applications*, 126(2), 2015.

[9] Chao Liu, Qian Lin, Hyun Kim, and Mark Yim. SMORES-EP, a Modular Robot with Parallel Self-assembly. *CoRR*, abs/2104.00800, 2021. URL https://arxiv.org/abs/2104.00800.

[10] Melinda Malley, Bahar Haghighat, Lucie Houe, and Radhika Nagpal. Eciton robotica: Design and algorithms for an adaptive self-assembling soft robot collective. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4565–4571. IEEE, 2020. URL https://ieeexplore.ieee.org/abstract/document/9196565.

[11] Nathan Melenbrink and Justin Werfel. Local force cues for strength and stability in a distributed robotic construction system. *Swarm Intelligence*, 12(2):129–153, 2018. URL https://link.springer.com/article/10.1007/s11721-017-0149-2.

[12] Nathan Melenbrink, Paul Kassabian, Achim Menges, and Justin Werfel. Towards Force-aware Robot Collectives for On-site Construction. In *ACADIA 2017:Disciplines & Disruption*, pages 382–391, 2017. URL http://papers.cumincad.org/cgi-bin/works/paper/acadia17_382.

[13] Nathan Melenbrink, Panagiotis Michalatos, Paul Kassabian, and Justin Werfel. Using local force measurements to guide construction by distributed climbing robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4333–4340. IEEE, 2017. URL https://ieeexplore.ieee.org/abstract/document/8206298.

[14] Francesco Mondada, Michael Bonani, André Guignard, Stéphane Magnenat, Christian Studer, and Dario Floreano. Superlinear physical performances in a SWARM-BOT. In *European Conference on Artificial Life*, pages 282–291. Springer, 2005. URL https://link.springer.com/chapter/10.1007/11553090_29.

[15] Benoît Piranda, Pawel Chodkiewicz, Pawel Holobut, Stéphane P. A. Bordas, Julien Bourgeois, and Jakub Lengiewicz. Distributed prediction of unsafe reconfiguration scenarios of modular-robotic Programmable Matter. *CoRR*, abs/2006.11071, 2020. URL https://arxiv.org/abs/2006.11071.

[16] Chris R Reid, Matthew J Lutz, Scott Powell, Albert B Kao, Iain D Couzin, and Simon Garnier. Army ants dynamically adjust living bridges in response to a cost–benefit trade-off. *Proceedings of the National Academy of Sciences*, 112(49):15113–15118, 2015. URL https://www.pnas.org/content/112/49/15113.short.

[17] John W Romanishin, Kyle Gilpin, Sebastian Claici, and Daniela Rus. 3D M-Blocks: Self-reconfiguring robots capable of locomotion via pivoting in three dimensions. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1925–1932. IEEE, 2015. URL https://ieeexplore.ieee.org/abstract/document/7139450.

[18] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014. URL https://www.science.org/doi/full/10.1126/science.1254295.

[19] Kasper Stoy, David Brandt, and David Christensen. *Self-Reconfigurable Robots: An Introduction*. MIT press, 2010.

[20] Petras Swissler and Michael Rubenstein. ReactiveBuild: Environment-Adaptive Self-Assembly of Amorphous Structures. In *International Symposium Distributed Autonomous Robotic Systems*, pages 363–375. Springer, 2021. URL https://link.springer.com/chapter/10.1007/978-3-030-92790-5_28.

[21] Niken Syafitri. *Self-organising assembly using swarm robots*. PhD thesis, University of Southampton, 2018. URL https://eprints.soton.ac.uk/418969/.

[22] Ritchie Vink. *anaStruct*. 2021. URL https://github.com/ritchie46/anaStruct.

[23] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014. URL https://www.science.org/doi/full/10.1126/science.1245842.