Distributed autonomous morphogenesis in a self-assembling robotic system

Wenguo Liu and Alan F. T. Winfield

1 Introduction

The EU-funded project SYMBRION (http://www.symbrion.eu) is aiming to develop a super-large-scale swarm of robots which is able to autonomously assemble to form 3D symbiotic organisms to perform tasks. The idea is to combine the advantages of swarm and self-reconfigurable robotics systems to investigate and develop novel principles of evolution and adaptation for robotic organisms from bio-inspired and evolutionary perspectives [7]. Each robot in such a system can either work autonomously or self-assemble into various morphologies when required, as shown in Figure 1. Unlike modular self-reconfigurable robotic systems such as PolyBot G3 [17], CONRO [11], M-TRAN III [9] and SuperBot [12], (see [3, 16] for a survey of such systems), in SYMBRION individual robots are independently mobile and will be able to autonomously aggregate and dock with each other. The robots will initially form a 2D planar organism. Once the robots in the 2D planar organism have assumed the correct functionality, according to their position in the organism, the organism will lift itself from 2D planar configuration to 3D configuration and, with respect to locomotion, will function as a macroscopic whole. The aggregated organism will also be able to disassemble and reassemble into different morphologies to fit the requirements of the task.

The morphologies of the organism that the robots can self-assemble into must be constrained by the specific hardware design of the individual robots. With only limited sensory capabilities, it is a challenge to coordinate the behaviours of a large number of robots in a decentralised manner in order that the robots can form some desired structures. A bio-inspired gradient based process has been widely used to to study the pattern growth problem in agent-based cell systems [10, 2, 15]. Various morphology control mechanisms have also been proposed for controlling different

Wenguo Liu and Alan F. T. Winfield
Bristol Robotics Laboratory, University of the West of England, Bristol, UK. e-mail: wenguo.liu@brl.ac.uk, alan.winfield@uwe.ac.uk

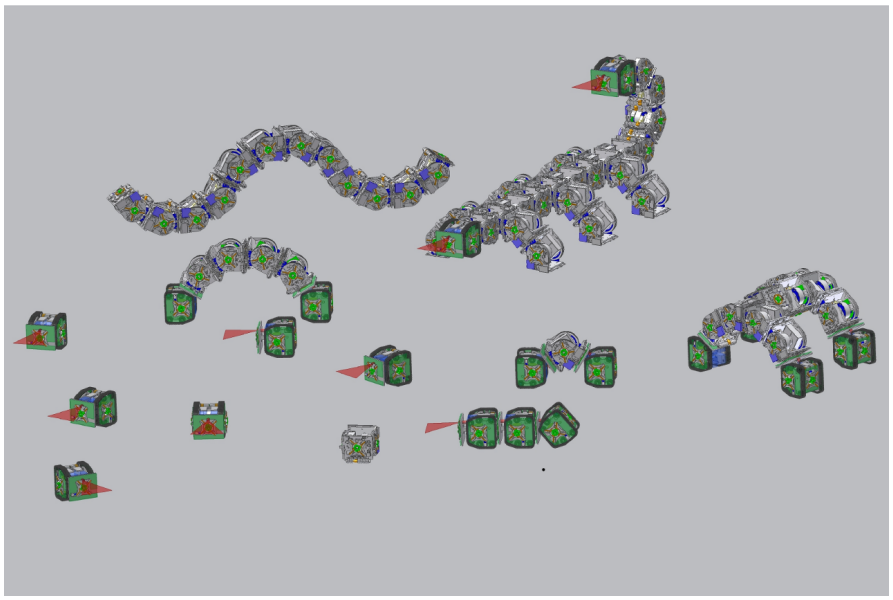


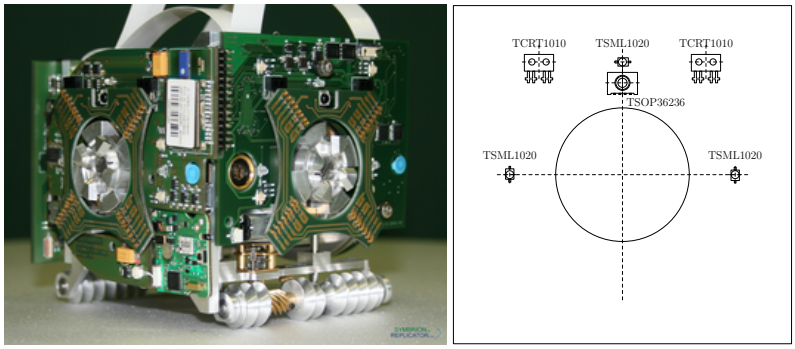
Fig. 1: A vision of the SYMBRION project. Two types of robots, with different motion capability but compatible mechanical docking units, are proposed to be developed in the system. All robots can explore the environment freely using their own sensing and actuators. Different structures can be formed when several robots physically connect to each other, such as a 'snake' like shape and a 'scorpion' like shape shown here.

modular robotic systems in recent years. Støy [13] has evaluated a gradient-based approach to control the self-reconfiguration of cubic units in simulation, where the desired configuration is grown from an initial seed module and guided by the gradient in the system using local communication. Guo et al [5] proposed a distributed gene regulatory network (GRN) based algorithm for multi-robot construction, in which the global shape information is embedded into the GRN dynamics directly and the local interaction among the robots is represented by the diffusion terms; they showed, in simulation, that different pre-defined simple shapes can be formed. Also tested in simulation, Grushin and Reggia [4] developed an automated rule generation procedure that allows structures to successfully self-assemble in an environment with constrained, continuous motion. Apart from controlling the morphologies of lattice type or chain type robots, Christensen et al. have proposed a simple language, SWARMMORPH-script, for arbitrary morphology generation for self-assembling robots [1], where each robot is fully autonomous. The morphologies are pre-specified as sets of rules stored in scripts which can be communicated and subsequently executed on the newly connected robot. Their morphology control algorithm has been demonstrated using a group of *s-bot* robots in a 2D environment. Our work also needs to consider the morphology control problem for a swarm of au-

tonomous mobile robots. However, this chapter focuses on how specific structures can be formed based on the existing sensing and communication capabilities of the SYMBRION robot.

This chapter is organised as follows: Section 2 introduces the robot platform used in this study. Section 3 presents the controller design for each single robot. This section gives a detailed discussion about the local communication protocols and the behaviours for the robots. Section 4 discusses two different recruitment strategies for growing an organism. Section 5 verifies the morphology control mechanism in simulation and Section 6 concludes the chapter.

2 The SYMBRION robots and their docking sensors



(a) (b)

Fig. 2: a) The first generation prototype of a SYMBRION robot and, b) the placement of the IR sensors on each vertical side PCB.

Table 1: Infra-red sensors for autonomous docking

Sensors	Quantity	Range	Purpose
Proximity sensor	8	15cm	obstacle and robot detection
Docking alignment sensor	8	25cm	IR beacon signal detection
IR communication channel	4	150cm	general communication duty and bearing detection

Figure 2(a) shows a first generation SYMBRION robot. It has a cubic shape sized 8cm x 8cm x 8cm. The robot can move omnidirectionally in a 2D planar environment using two screwdrive type wheels, and bend 90 degrees along the common axis of two opposite docking units using a hinge drive, which is in parallel with the wheel

axis. A rich set of sensors are proposed to be installed in the robot for environmental perception, locomotion and internal state monitoring purposes, see [6] for a full list. Four mechanical docking units, one on each vertical side, are installed on the robot to allow stable physical connections between robots. In addition, electrical contacts next to the docking units can be coupled automatically to provide inter-robot communication and power sharing busses between two connected robots. The docking units can handle misalignment in horizontal and vertical directions as well as rotation within certain ranges. This configuration gives the robots flexibility on either working as fully autonomous units with their own perception and actuation capability, or as a whole organism sharing the sensing, computation and energy cross the busses.

To achieve autonomous docking in a 2D planar environment, specific infrared (IR)-based sensing – including proximity detection and docking alignment detection – and local communications circuits have been developed for the SYMBRION robot, see [8]. As shown in Table 1, Each robot is endowed with 8 proximity sensors, 8 docking alignment sensors and 4 channel local communications for autonomous docking, the maximum detection range for each function is about 15cm, 25cm and 150cm respectively. These sensors have the same placement on each side PCB of the robot, as shown in Figure 2(b). More specifically, two IR sensors (TCRT1010) have been placed symmetrically above and on either side of the docking unit (marked with a circle); one IR LED (TSML1020) is placed directly above the docking unit, while the other two LEDS are located on either side of the docking unit. These LEDs are used to emit different frequency signals for obstacle detection, docking alignment and communication. The IR sensors work for both obstacle detection and docking alignment detection. As for communications, one IR remote control receiver (TSOP36236) is placed next to the IR LED on each side PCB. Note that the 4 channels of local communication can work simultaneously. By default they are all in "listening" mode; whenever one robot is broadcasting messages, another robot within range will receive the message with one or two adjacent channels, which provide the robot with an approximation of the direction of the signalling robot.

3 The Controller framework for autonomous morphogensis

Based on the sensing capability of the SYMBRION robots, the autonomous docking approach can be illustrated as in Figure 3: once one robot (A) in the swarm decides to initialise the docking process, it will broadcast some recruitment signals via specified communication channels to attract other robots. The signals can be detected by other robots within range (150cm) to provide rough direction information to the recruiting robot. For example, robot (B) in Figure 3 detects the recruitment signals and hence moves toward robot (A) along the direction detected. As the recruitment signals alone are not enough to guide the docking approach (they are in fact repeatedly broadcast messages as explained later), the recruiting robot meanwhile emits some fixed frequency IR signals, namely beacon signals, on the docking faces where new

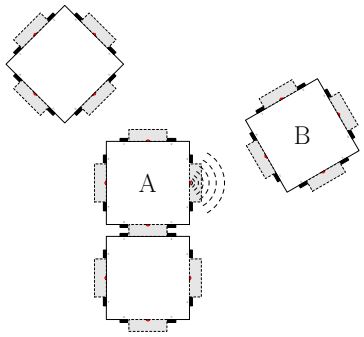


Fig. 3: A scenario of autonomous docking. Robot A is emitting some signals to recruit another robot. Robot B that detects the signals is trying to align with Robot A for docking.

robots need to be recruited. The beacon signals can only be detected by other robots at short range (15cm). They are used to guide the approaching robots to execute precise alignment towards the docking face. Once two robots are well aligned and close enough, a physical docking/locking process can be started. Upon the completion of the locking process, the recruiting robot stops emitting beacon signals. To form required 2D structures, new robots in the partially assembled organism will be selected to recruit more robots following some rules. The same process is repeated until the specified structure is formed. Thereafter, the robots in the organism must determine collectively whether the current structure is suitable for the task. If not, a new shape must be selected; all or some of the robots must disconnect from the organism and a new cycle of self-assembly started until the organism can achieve its goal. A behaviour-based approach is adopted for the design of the morphogenesis controller as described in the following sections.

3.1 A finite state machine

Figure 4 shows the finite state machine (FSM) for the morphogenesis controller. According to the physical connection status of the robot, the 8 states in the FSM can be grouped into two blocks as marked with dashed lines in Figure 4 – *swarm mode* and *organism mode*. Switching between these two modes occurs whenever a robot either docks with or undocks from another robot in the organism. For the robots in *organism mode*, the default state is *InOrganism*; this may change to state *Recruitment* or *Disassembly* during the self-assembly process and transitions are determined by the morphogenesis strategy applied by robots. Once robots are in state *Recruitment*, they will flash some of their IR LEDs – the docking beacon – to attract other robots in *swarm mode* to dock. For the robots in *swarm mode*, the default state is *Flocking*, which is, here simply, a place holder for all other swarm mode behaviours, not as-

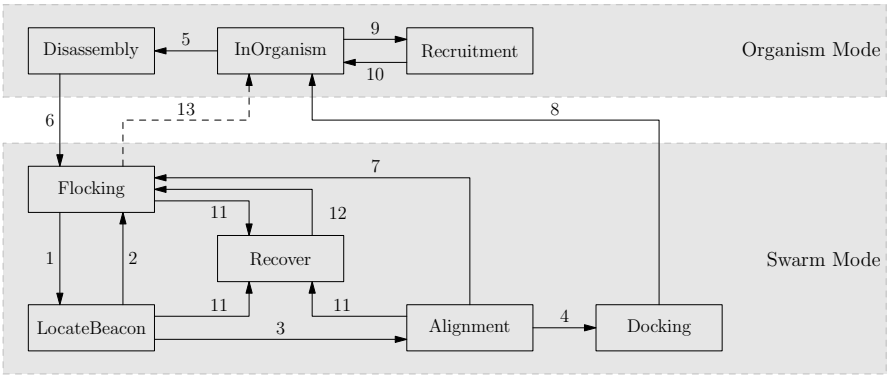


Fig. 4: Robot finite state machine (FSM) for the autonomous morphogenesis controller

Table 2: Conditions causing state transitions

Number	Description
1	docking message received
2	collision, or no docking message received
3	docking beacon signals detected
4	aligned and ready to dock
5	disassembly required
6	undocking completed
7	expelling message received, or docking signals lost
8	docking completed
9	recruitment required
10	recruitment completed
11	robot stalled
12	recover done
13	decision to become a seed robot

sociated with self-assembly or disassembly. As indicated with a dashed line, a robot in state *Flocking* may transfer to state *InOrganism* and then to state *Recruitment* to start the self-assembly process – this is the seed robot (see section 3.3). Although only one seed robot can occur within one growing organism multiple seed robots may co-exist in the swarm.

In general, for all robots in *swarm mode*, when a recruitment signal is sensed they will move towards it and try to dock the recruiting robot; here transitions from one state to another are triggered by the combination of IR sensing and communication. Table 2 lists all of the conditions which cause state transitions in the FSM. Note that a *Recover* state is introduced to deal with robots that become stalled. This could happen, for instance, because of corner collisions or blind spots in robot's collision sensors.

3.2 Behaviours

Communication plays a crucial role on coordination of the behaviour for the robots when self-assembly is in progress. In the SYMBRION robots, two kinds of local communication are proposed to fulfil this purpose. When robots are in *swarm mode*, IR-based communication is used to self-organise the autonomous docking process. At this stage, robots simply broadcast some simple message tokens when required. Note that when transmitting messages, only one or two specific communication channels are used. Since the IR signals may be occluded and have a certain transmission angle and range, the number of candidate receivers is limited, as we would expect. To reduce the complexity of the communication protocols, five fixed message tokens, each of 1-Byte length, are broadcast by the robots when communication is required, as follows:

- *MSG-Recruitment* is to indicate that a recruitment process has started. The message is broadcast and repeated by the robots in state *Recruitment*. It is used by other robots to locate the direction of a recruiting robot in longer range with less accuracy.
- *MSG-InRange* is transmitted by the robot in state *LocateBeacon* when it detects beacon signals (transmitted by one of the IR LEDs of a recruiting robot). The message is used to inform the recruiting robot to stop transmitting MSG-Recruitment messages.
- *MSG-Expelling* is broadcast by the robot in state *Alignment* to expel other competitors in order to make more room for docking alignment and thus reduce interference.
- *MSG-DockingReady* is sent by the robot in state *Docking* when its docking unit is fully in position to the recruiting robot. It is used to inform the recruiting robot to stop emitting beacon signals and start to lock the docking units.
- *MSG-UnDocked* is sent by the robot in state *Disassembly* when the undocking procedure is fully completed. The robot which was previously docked will receive this message.

For robots in *organism mode*, communication is implemented through the common bus between two coupled docking units. Apart from the coordination and controlling of behaviour for the whole organism, which is beyond the scope of this study, the autonomous morphogenesis also requires sharing information among the partially assembled organism. To implement the recruitment strategies discussed later, the following essential information needs to be shared via the communication bus.

- Notification when new robots join the organism. A *MSG-NewRobotAttached* message will be sent by the recruiting robot when a new robot is docked. The message is then propagated by every docked neighbour robot in the organism. It is used to trigger the transitions between states *InOrganism*, *Disassembly* and *Recruitment* along with the recruitment strategies explained later in section 4.

- Information about the current organism structures. The newly docked robots need to acquire this information from the recruiting robots such as the number of robots in the partially assembled organism, the final shape of the organism, etc. However, the message content will vary when different recruitment strategies are applied.

The behaviours of each state of the FSM can be defined as follows:

- InOrganism Robot remains static in the organism while monitoring the communication busses. When a *MSG-NewRobotAttached* message is received from one of the channels, it checks whether it needs to switch to state *Recruitment* or *Disassembly* following certain rules. Then it sends the *MSG-NewRobotAttached* messages to other docked neighbour robots, excluding the one it received the message from.
- Recruitment Robot chooses one or several docking faces, based on the recruitment strategy, from which to emit beacon signals and *MSG-Recruitment* messages at the same time. Once it detects a *MSG-InRange* message, it stops transmitting *MSG-Recruitment* to avoid attracting too many robots. The robot performs a mechanism docking lock when the *MSG-DockingReady* message is received. It then moves to state *InOrganism* and send *MSG-NewRobotDocked* messages to all connected robots.
- Disassembling Robot executes an action sequence to undock from the organism if only one of its docking units is connected. It then sends a *MSG-UnDocked* message to the robot previously connected and moves to state *Flocking*. If more than one docking units are connected, it continues to wait.
- Flocking Robot wanders in the environment and searches for docking beacons. It avoids obstacles and other robots. When *MSG-Recruitment* messages are received it moves to state *LocateBeacon*. This state is also a place holder for any other behaviours in *swarm mode* which are not relevant to the self-assembly process. For example, robots in state *Flocking* can perform work and under certain conditions become seed robots.
- LocateBeacon Robot approximately locates the beacon using 4 IR communication channels and moves in the direction of the beacon signals. If no *MSG-Recruitment* messages are received, or obstacles are detected, it transfers back to state *Flocking*. If beacon signals are detected, it sends a *MSG-InRange* message and then moves to state *Alignment*.
- Alignment Robot adjusts its headings and tries to minimise the misalignment of two docking units. It transmits *MSG-Expelling* messages repeatedly to expel competitors. However, if it detects *MSG-Expelling* messages from other robots, it exits to state *Flocking*. Once two docking units are aligned and close enough (based on readings from the beacon detection sensors and proximity sensors), it transmits a *MSG-DockingReady* message and moves to state *Docking*.
- Docking Robot performs a mechanical docking procedure to physically connect to the organism. It moves to state *InOrganism* upon completion.

- Recover Whenever robot gets stalled with obstacles or other robots, it executes some action sequence to rescue itself. If succeeded, it moves to state *Flocking* regardless its previous state.

3.3 The seed robots

The seed robot is defined as the robot that initiates the process of self-assembly for an organism. It is the first robot in the organism and is not recruited by other robot. Clearly, as all robots in the system have the same controller, any of them can choose to become a seed robot when certain conditions are satisfied. The seed robot will decide the initial shape of the organism the robots need to self-assemble into. It has the duty of monitoring the self-assembly process and notifying the completion of the construction within the formed organism. In case reshaping is required, the seed robot is also in charge of initiating the reshaping procedure. Note that multiple seed robots can co-exist in the overall system since multiple organisms may arise at the same time. However, each organism can only have one seed robot. A detailed explanation of how robots choose to become seed robots is outside the scope of this chapter, however, we anticipate that some environmental feature such as a wall judged by the robot too high to go around would trigger this transition.

3.4 Competition resolution

Competition may arise when multiple robots detect the recruitment signals that emit from the same source at the same time. Without any competition resolution mechanism, these robots will be attracted to the recruiting robot, gradually surrounding it. Clearly, only one of them can dock with the recruiting robot. To reduce the competition in this case, two levels of simple competition resolution mechanisms has been applied. First, the recruiting robot will stop broadcasting recruitment messages whenever any robot detects the beacon signals (and responds by sending *MSG-InRange* to the recruiting robot), this prevents more robots being attracted to the recruiting robot. Secondly, the robot that detects the beacon signals (in state *Alignment*) will broadcast *MSG-Explelling* messages via left and right channels to expel its competitors. In addition, as the recruiting robot may need to open more than one docking faces to recruit robots, the recruitment messages via different channels are broadcast at different intervals to avoid confusing the receivers.

4 Recruitment strategies

To grow a specific organism shape from one seed robot, the right strategies have to be applied. In other words, the robots in the partially assembled organism must determine the location and timing at which a new robot needs to be recruited and connected. More specifically, the following questions need to be addressed:

- which robots need to move from state *InOrganism* to state *Recruitment*;
- which docking faces the recruiting robot needs to open to recruit new robots.

These problems are referred as recruitment strategies in this study. As there is no central control unit in the partially assembled organism, each robot in the developing organism have to autonomously decide the next steps in the self-assembly process.

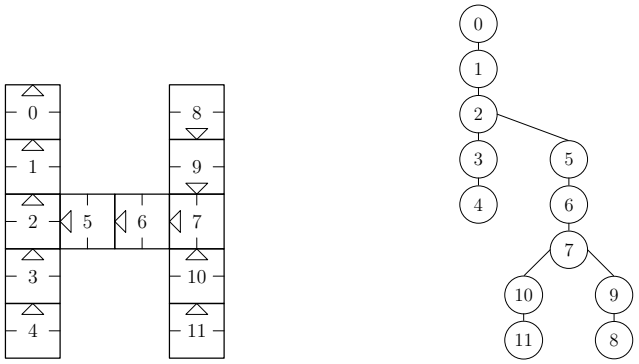
4.1 Representation of organisms

Before we can address the recruitment strategy a common representation for the pre-defined organism structures must be defined. During an autonomous docking process a recruiting robot is normally static while emitting the docking beacon signals. Although each robot has four side docking units named front, left, back and right, the locomotion capability of a single robot dictates that robots will use their front side only to dock onto the recruiting robot. Therefore, for any connection between two docking units in the organism, one and only one front side docking unit must be present. If each robot in the organism is treated as a node in a tree data structure where the "parent", "lchild", "mchild" and "rchild" of the node represent the front, left, back and right side of a robot respectively, as shown in Figure 5, then the whole organism in a 2D planar environment can be represented as a tree data structure in which each edge denotes a physical docked connection between two robots.

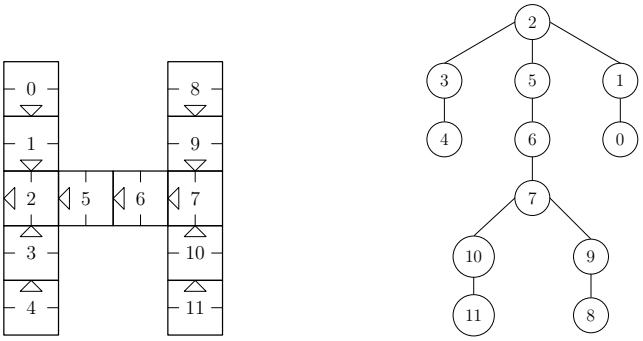
Fig. 5: A robot and its graphical node counterpart. The hinge joint is on the left-right axis.

Figure 6(b)(c) show two organisms and their corresponding tree data structure representations. Although these two organisms have very similar 2D structures, because of the orientation of the hinge driver of the robots (marked with two line segments from the left and right sides of a robot in Figure 6), they will have different 3D locomotion capabilities. Therefore, their tree data structure representations

are complete different. Clearly, the start point for self-assembly of an organism, i.e. the seed robot, cannot be arbitrarily chosen. It must be the root node of its corresponding tree representation. In the following sections, two different recruitment strategies will be investigated based on this graphical representation.



(a) Organism 1



(b) Organism 2

Fig. 6: Graphical representation of organism structures. Although two organisms have very similar 2D planar structures, the orientation difference for robots with ID '2' leads to different 3D motion capability for these two organisms, when those robots bend their hinge joint.

4.2 Strategy 1 - single entry recruitment

As IR signals are used for recruitment and docking alignment, interference may arise if more than on light source are actively emitting IR signals for this purpose, at the same time. The first strategy is to allow only one of the robots in the organism

to move into state *Recruitment* with only one docking face activated, at any one time. In other words, only one robot can be recruited to the organism every time – a single entry recruitment. Thus, the order in which robots attach to the organism can be retrieved by a pre-order walk of its tree representation. Take organism 1 shown in Figure 6(a) as an example, assume each node has been identified with an unique name, if the children of a node are visited in the order "mchild – lchild – rchild", then the robots can be recruited to the organism in the order of list {0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 9, 8}, named *sortedNodeList*, where the first robot, No. 0, will act as a seed robot. Other robots in the *sortedNodeList* are recruited by their parent node one by one. The order that the robots move into the *Recruitment* state is in fact the order of the parent nodes of each node in the pre-order walk node list, i.e. {0, 1, 2, 3, 2, 5, 6, 7, 10, 7, 9} for organism 1. The recruitment side of each recruiting robot can also be easily retrieved from the tree representation. If we introduce an ordered pair "(Robot-ID, Recruitment-Side)", then to grow organism 1, the order that the robots move to state *Recruitment* and their corresponding recruitment sides can be expressed as list {(0, 0), (1, 0), (2, 0), (3, 0), (2, 2), (5, 0), (6, 0), (7, 1), (10, 0), (7, 2), (9, 0)}, named *recruitmentNodeList*, where number 0, 1, 2 in the second element of each pair denote the Back, Left and Right side of a robot respectively. Similarly, for organism 2, *sortedNodeList* = {2, 5, 6, 7, 10, 11, 9, 8, 3, 4, 1, 0}, and *recruitmentNodeList* = {(2, 0), (5, 0), (6, 0), (7, 1), (10, 0), (7, 2), (9, 0), (2, 1), (3, 0), (2, 2), (1, 0)}. Listing 1 shows an algorithm to generate these two node lists recursively from the tree data structure of an organism.

Algorithm 1 Generating two node lists recursively

```
Precedure: PreOrderTraversal (node)
Input: the root node of a tree
 1: if node ≠ φ then
 2:     sortedNodeList.push(node→id);
 3:     if node→GetParent() ≠ φ then
 4:         RecruitmentNode rnode;
 5:         rnode.id ⇐ node →GetParent()→id;
 6:         for i = 0 to 2 do
 7:             if node == node→GetParent()→GetChildByIndex(i) then
 8:                 rnode.side ⇐ i;
 9:                 recruitmentNodeList.push(rnode);
10:             end if
11:         end for
12:     end if
13:     PreOrderTraversal (node→GetMiddleChild());
14:     PreOrderTraversal (node→GetLeftChild());
15:     PreOrderTraversal (node→GetRightChild());
16: end if
```

For any specific tree structure, a pair of *sortedNodeList* and *recruitmentNodeList* give sufficient information for the swarm to self-assemble into the corresponding 2D organism structure. Listing 2 depicts the controlling code in states *InOrganism* and

Recruitment. Note that the robot in the organism will be allocated with unique ID only when it docks to the organism. Its identity is reset to null whenever it undocks from the organism. We also assume that all robots store the same information about the organisms, i.e. pairs of *sortedNodeList* and *recruitmentNodeList*.

Algorithm 2 Single entry recruitment strategy for robots in *organism mode*

Behaviour: in state *InOrganism*
1: **if** a *MSG_NewRobotAttached* received **then**
2: propagate the *MSG_NewRobotAttached* to connected neighbours, except the one it received the message from;
3: `num_robots_inorganism` $\Leftarrow$ `num_robots_inorganism` + 1;
4: **end if**
5: **if** its ID matches with the ID from *recruitmentNodeList* which is indexed by num_robots_inorganism **then**
6: enable the corresponding docking channel and start to emit beacon signals;
7: move to state *Recruitment*;
8: **end if**
Behaviour: in state *Recruitment*
9: **if** a *MSG_DockingDone* received from the recruiting channel **then**
10: send information about the current developing organism to newly docked robot;
11: propagate the *MSG_NewRobotAttached* to connected neighbours, except the newly docked one;
12: stop emitting beacon signals;
13: `num_robots_inorganism` $\Leftarrow$ `num_robots_inorganism` + 1;
14: move to state *InOrganism*;
15: **else if** a *MSG_InRange* received from the recruiting channel **then**
16: stop broadcasting *MSG_Recruitment*;
17: reset internal timer T_1;
18: **else if** T_1 is up **and** it is the time slot for sending recruitment signals **then**
19: broadcast *MSG_Recruitment* via corresponding recruiting channel;
20: **end if**

Take organism 2 as an example, the recruitment strategies are described as follows: the seed robot first retrieves its ID from the *sortedNodeList* and the recruitment side from the *recruitmentNodeList*, where ID = 2, side = 0 (Back). It then starts to emit *MSG-recruitment* messages and docking beacon signals to recruit other robots. When a new robot is docked to its Back side, it sends a message to this robot with the index of the organism and how many robots are in the organism; here index is 2 (corresponding to organism 2) and the number of robots in the organism is 2. The newly docked robot then retrieves its ID from the corresponding *recruitmentNodeList*, here 5 as it knows it is the second robot in the organism. These two robots then move to state *InOrganism*, where they compare their IDs with the ID of the second pair element in the *recruitmentNodeList*. Since it is "(5, 0)", the robot in the organism with ID "5" moves into state *Recruitment* with side 0 (Back) to attract another robot. Similarly, the newly docked robot will receive the index of the organism and the current number of robots in the organism from robot "5", it is then assigned an ID of "6". Meanwhile, robot "5" will propagate a *MSG-NewRobotDocked* message via its Front side. Robot "2" receives this message and will increment its internal vari-

able `num_robots_inorganism` by 1, now 3. Next, robot "6" in state *InOrganism* will be matched as the recruiting robot from the *recruitmentNodeList*. The process continues until all robots' internal variable `num_robots_inorganism` is equal to the size of the *sortedNodeList*.

4.3 Strategy 2 - multiple entries recruitment

The second strategy for growing an organism from one seed robot is to allow multiple robots to be recruited to the organism at the same time, notwithstanding the potential interference and competition that may arise due to the multiple IR light sources. The idea is to activate all docking faces of the developing organisms for docking where new robots need to be recruited. This implies no limitation will be applied on the number of robots moving to state *Recruitment* and on the number of docking units that one robot is allowed to open for recruiting new robots. As the robots in the *swarm mode* behave independently, the order that the robots are recruited to the organism are unlikely to remain the same in subsequent self-assembly of the same organism. To implement the idea of multiple entries recruitment, how the organism structures are stored in the robot and are transferred between robots needs to be addressed first. Unlike the ID-based node lists introduced in the first strategy, here the tree representation of an organism will be described with a well-formed parenthesis symbol sequence over {'B', 'L', 'R', '0'}, where 'B', 'L' and 'R' stand for "back", "left" and "right" respectively. The symbol sequence is annotated with a nested structure, corresponding to the edges of the tree. The length of a symbol sequence is two times the number of edges of the tree, of which half are filled with character '0'. Figure 7 shows an example of a 2D organism structure and its symbol sequence.

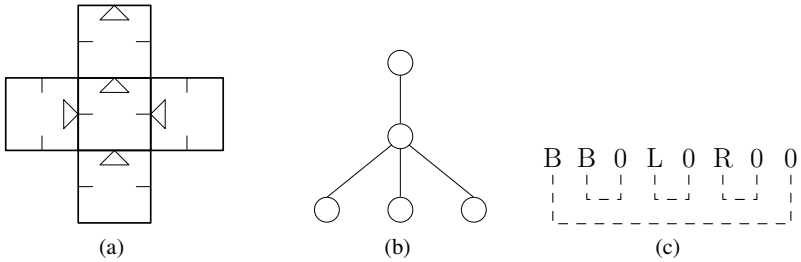


Fig. 7: (a) A simple 'cross' 2D organism, (b) its tree representation and (c) the symbol sequence notation.

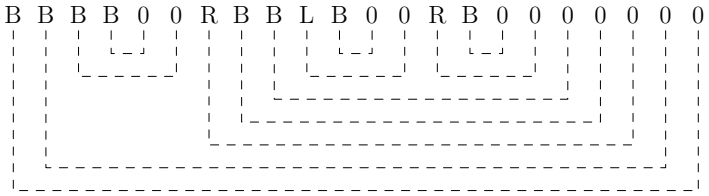
The symbol sequence can be obtained by performing a pre-order traversal of the tree, adding a 'B', 'L', or 'R' (depending on which child node the edge is connected to) each time an edge is first traversed and adding a '0' each time an edge is traversed

in the opposite direction. A recursive algorithm for obtaining the symbol sequence from the tree representation of an organism is shown in Listing 3.

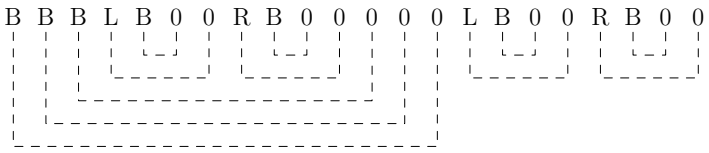
Algorithm 3 Generating the symbol sequence recursively

```
Precedure: PreOrderTraversal (node, side)
Input: a tree node and one of its child side (∈ {MIDDLE, LEFT, RIGHT})
 1: if node ≠ φ then
 2:     if node→GetParent() ≠ φ then
 3:         symbolSequence.push(GetSymbolBySide(side));
 4:     end if
 5:     PreOrderTraversal (node→GetMiddleChild(), MIDDLE);
 6:     PreOrderTraversal (node→GetLeftChild(), LEFT);
 7:     PreOrderTraversal (node→GetRightChild(), RIGHT);
 8:     if node→GetParent() ≠ φ then
 9:         symbolSequence.push('0');
10:     end if
11: end if
```

Figure 8 then shows two symbol sequences for the organism 1 and 2 introduced in previous section. Clearly, the symbol sequence notation yields a very compact for-



(a) organism 1



(b) organism 2

Fig. 8: Symbol sequences of example organisms. The dashed line that connects two symbols indicates a corresponding edge in its tree representation counterpart.

mat for storing and transferring an organism structure among the robots. It is very computationally efficient to compare two different organism structures by comparing two symbol sequence strings. In addition, extracting or searching for the sub-organism from complex shapes can also be easily implemented.

Similar to the tree representation of section 4.1, a symbol sequence may include up to three branch[1] sequences, namely left, back and right branch respectively. For example, organism 1 has only one back branch **BBBB00RBBLB00RB0000000** while organism 2 has a back branch **BBBLB00RB00000**, a left branch **LB00** and a right branch **RB00**. Along with the symbol sequence representation, an implementation of the multiple entries recruitment strategy is shown in Algorithm 4.

Algorithm 4 Multiple entries recruitment strategy for robots in *organism mode*

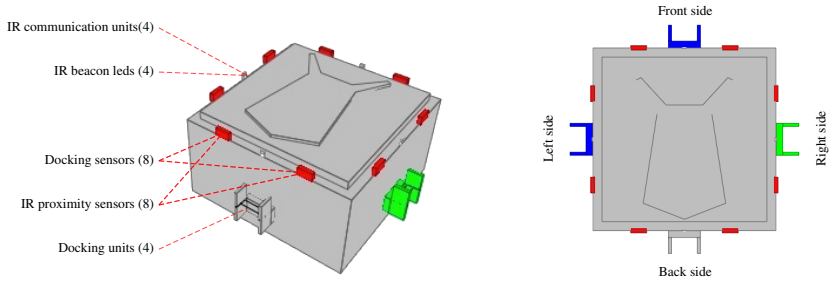
Behaviour: in state *InOrganism*
1: **if** new symbol sequence information received **then**
2: extract the branches from the symbol sequence;
3: **for** i=0 **to** 2 **do**
4: **if** GetBranch(i) $\neq \phi$ **then**
5: enable the corresponding docking channel and start to emit beacon signals;
6: **end if**
7: **end for**
8: move to state *Recruitment*;
9: **end if**
Behaviour: in state *Recruitment*
10: **for** i = 0 **to** 2 **do**
11: **if** a *MSG_InRange* received from channel i **then**
12: stop broadcasting *MSG_Recruitment*;
13: reset the internal timer T_i;
14: **else if** T_i is up **and** it is the time slot for sending recruitment signal **and** the corresponding recruiting channel is not connected **then**
15: broadcast *MSG_Recruitment* via corresponding channel;
16: **end if**
17: **if** a *MSG_DockingDone* received from channel i **then**
18: stop emitting beacon signals from channel i;
19: send a branch symbol sequence, if it exists, to corresponding newly docking robot;
20: **if** all required docking faces are connected with new robots **then**
21: move to state *InOrganism*;
22: **end if**
23: **end if**
24: **end for**

When the strategy is applied, the newly docked robot may receive a new symbol sequence from the recruiting robot it is docked to. If yes, this robot moves to state *Recruitment* and enables the docking faces where the corresponding branch sequences exist. For the robot in state *Recruitment*, whenever a new robot docks to it, it sends a symbol sequence, which is generated by removing the first and the last characters of the corresponding branch sequence, to the newly docked robot. Note that the transferred symbol sequence can be an empty string if the corresponding branch includes only one edge. Once all required docking faces are docked with new robots, it moves to state *InOrganism*. The process repeats until the organism is formed. Note that in this case only the seed robot has the completed symbol se-

[1] the subtrees of the root node

quence which represents the final organism structure, while all other robots in the organism store only sub sequences which include the robot itself and its children nodes. For clarification, take organism 2 as an example: The seed robot (robot "2") first moves to state *Recruitment* with its left, back and right docking faces enabled to recruit new robots, since its symbol sequence has one left, one back and one right branch. When a new robot (robot "5") docks to its back side, the seed robot will send a symbol sequence **BBLB00RB0000** to robot "5". Robot "5" then moves to *Recruitment* state with the back side docking faces enabled as there is no left and right branch in the symbol sequence of **BBLB00RB0000**. When robot "6" docks to robot "5", it gets a symbol sequence of **BLB00RB000**. Again, robot "6" enables its back side docking face only for recruiting new robots. Similarly, robot "7" gets a symbol sequence of **LB00RB00** from robot "6" and recruits new robots with its left and right docking face, robot "9" and "10" get **B0** from robot "7", robot "1" and "3" get **B0** from robot "2". When robot "4" docks to robot "3", it gets an empty symbol sequence from robot "3" hence stays in state *InOrganism*. The same rule applies for robot "0", "8" and "11". Note that this strategy does not require the robot in the organism to be allocated with a unique name, which is another advantage of the multiple entries recruitment strategy.

5 Verification in simulation

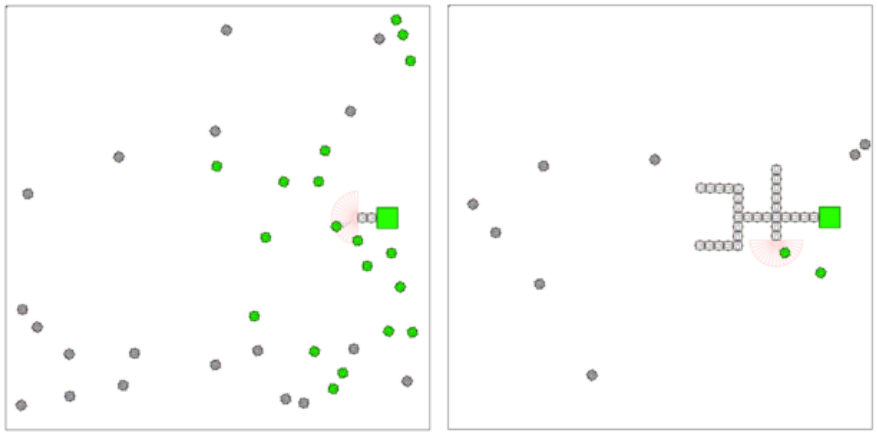


(a) perspective view (b) top-down view

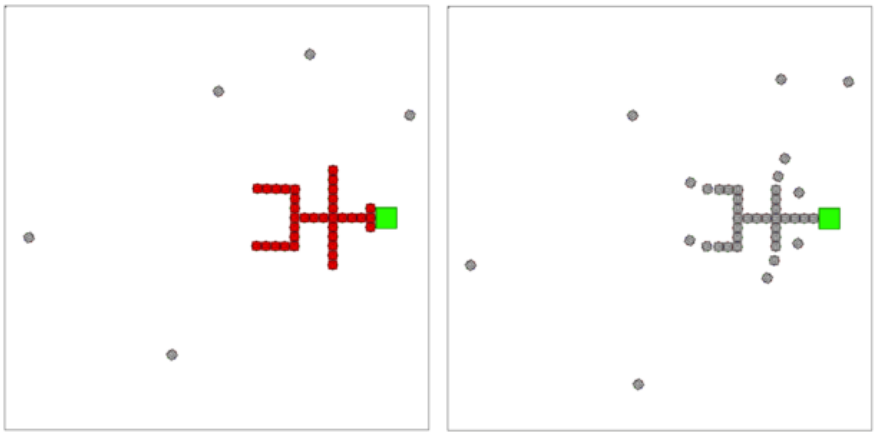
Fig. 9: A 2.5 dimensional SYMBRION robot model in Stage simulation.

At the time of writing the SYMBRION robot is still under development and not enough real robot platforms are available for testing the morphogenesis approach presented in this study. Thus a simulated model of the SYMBRION robot has been implemented in the popular simulation tool Stage [14]. As shown in Figure 9, the robot model in Stage has the same size as the SYMBRION robot. For each robot in Stage, the IR-based sensing and communications approach described in [8] is accurately simulated and calibrated with data measured from real sensors. Each

robot can move in the arena using two differentially driven wheels (not shown in Figure 9). Four simplified docking units on each vertical face of the robot simulate mechanical docking. As the morphogenesis approach discussed in the study takes place exclusively in a 2D environment, neither the hinge driver of the robot nor the physics needs to be simulated.



(a) 45m 41s 700msec (b) 1h 18m 24s 500msec



(c) 1h 32m 47s 500msec (d) 1h 33m 01s 500msec

Fig. 10: Screenshots from simulation using single entry strategy, the first robot is attached to the large box power socket at time 42m 52s 600msec. The organism is completed at time 1h 32m 30s 600msec.

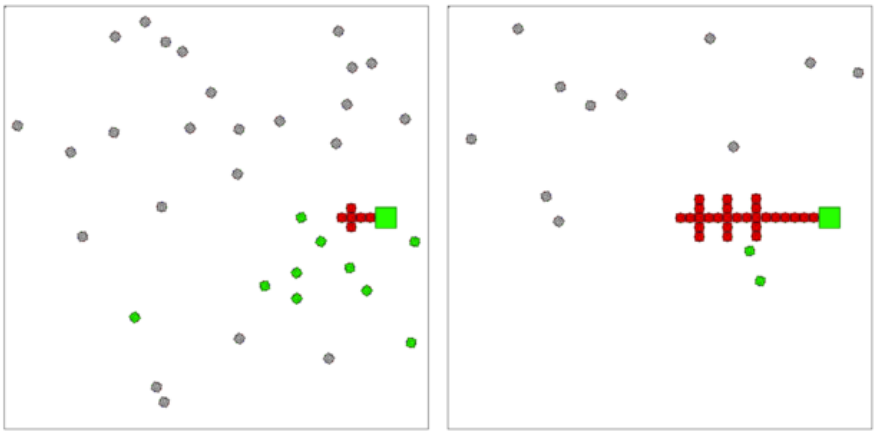
Simulation experiments are carried out within an 8 m $\times$ 8 m bounded arena. 40 robots are deployed, each running the same controller described in previous sections. Figure 10 shows screenshots from the Stage simulation in which the robots

are self-assembling into a complex 2D shape with 4-way and 3-way joints, and right angles, using the single entry recruitment strategy. To trigger the start of the morphogenesis process a large box acting as a "power socket", emitting IR signals which can be detected by the docking sensors of a robot, is placed in the arena. The first robot that finds the box becomes the seed robot and docks with the box. It then chooses, at random, one organism shape from its set of pre-defined structures and executes the recruitment strategy described in previous sections to recruit other robots and hence initiate the new structure. To further test the controller, once the organism has completely formed (Figure 10(c)), all robots in the organism are switched to state *Disassembly* for re-shaping. Figure 10(d) shows that the organism has started disassembling. After all robots are disconnected from the organism, the "power socket" starts to transmit IR signals again and the cycle is repeated. Each time, the seed robot randomly chooses a pre-defined organism and starts the recruitment procedure. Figure 11 shows some different 2D structures the robots have constructed within one single simulation run. As the IDs of the robots in the organism are dynamically allocated when they dock, the particular individual robots that make up the organism vary each cycle. Thus the same robot may play different roles, depending on its position, in different organisms.

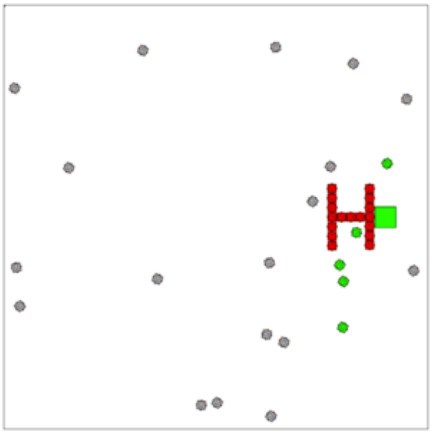
The same experiments have been performed using the multiple entry recruitment strategy. Figure 12 shows screenshots from the simulation in which the robots are trying to self-assemble to the same shapes as before. Clearly, this strategy allows more than one robot to recruit new robots into the developing organism. These are marked with dark colours in Figure 12. The order in which the robots move to state *Recruitment* depends on the shapes and the progress that the organism grows. As parallel docking from different positions is enabled, the organism grows much faster than with single entry recruitment strategy: the completion time is now around 19 minutes compared to 50 minutes.

6 Discussion and conclusion

Along with a commom behaviour-based controller framework, two recruitment strategies have been presented in this chapter for robots to self-assemble into 2D planar organisms. Both strategies are inspired from the tree representation of the 2D planar organism. The first strategy allows only one robot to be recruited into the organism at a time while the second strategy allows parallel docking. Accordingly, the organisms are represented and stored in the robots in different forms. Both are in compact formats, either arrays or strings, which is suitable for exchanging among robots via any communication means. The first strategy uses two ID-based node lists to represent the order that the robots join the organism and the order that the robots move to state *Recruitment*. To grow an organism, all robots in the system are required to store the same information about the organism. Each robot in the organism is dynamically allocated a unique ID when it docks to the organism. The ID is essential for the implementation of the recruitment strategy. Although there



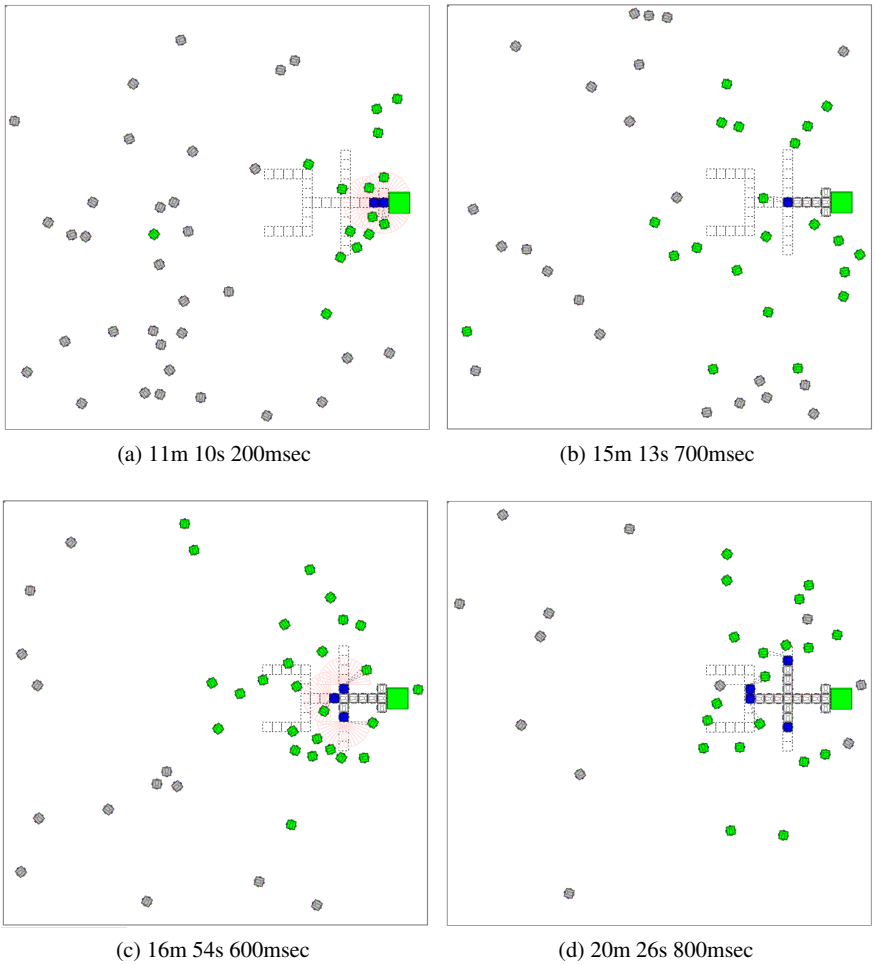
(a) a simple shape (b) a six leg structure with tail



(c) a four leg 'H' structure

Fig. 11: A selection of different 2D planar structures formed in simulation using single entry recruitment strategy.

is one-to-one mapping between the tree representation and the ID-based node lists, these two node lists do not explicitly store the structure information of an organism. In the second strategy, the organism is presented as a well-formatted symbol sequence. Similar to the ID-based node lists, a symbol sequence is generated by a pre-order walk in its corresponding tree representation. Each robot in the organism, except those with only the FRONT side docked (i.e. the leaf node in its tree presentation) and the seed robot, receives a sub symbol sequence from the robot it docks to (connected with FRONT docking face). The symbol sequence of a robot in the organism has direct information about the structure of the sub organism which in-



(a) 11m 10s 200msec (b) 15m 13s 700msec

(c) 16m 54s 600msec (d) 20m 26s 800msec

Fig. 12: Screenshots from simulation using multiple entries recruitment strategy. The robots marked with dark colour are in state *Recruitment*. The first robot is attached to the large box at time 10m 30s 100msec. The organism is completed at time 29m 38s 300msec.

cludes the robot itself and all its descendants nodes. Therefore, there is no need to use an unique name to identify each robot in the organism.

Both strategies have been validated using simulated SYMBRION robots in the Stage simulation, with the same sensing and communication capabilities for docking and recruitment as those in real robots. When very simple disassembly strategies are applied, re-shaping between different organisms can also be achieved using the same controller framework. Given the hardware constraints, in a 2D environment, the shapes can be any of those defined in tree structures with fewer than 3 children

and no cycles. Not surprisingly, as shown in the simulation, less time is required to develop an 2D organism when the second strategy is applied so that parallel docking is enabled. However, the gap between completion time, when different strategies are applied, may vary with the organism shapes the robots are trying to form. For example, it will take very similar time for the robot to grow a snake shape organism no matter which strategy is used, as in both cases only one robot can be recruited into the organism at a time. Although the system with the multiple entries strategy can, in general, achieve better performance in term of time efficiency towards developing a specific organism structure, higher interference and competition can emerge, resulting in a longer average time to recruit one robot from one specific location of the developing organism. Thus, the right strategy needs to be chosen carefully considering the performance metric of greatest concern.

In conclusion, the controller framework and the morphological control mechanism presented in this study leads to a fully distributed approach towards autonomous morphogenesis in the proposed self-assembly robotic system. Although this study considers only the scenario that the robots are constructing 2D planar organisms. The work will be expanded in the direction of both the *swarm mode* and *organism mode* using the same framework. For example, only very simple disassembly strategies have been implemented in this work. To improve the energy efficiency of the re-shaping procedure, more complex disassembly strategies need to be investigated in future work. Note also that at the time of writing the algorithm is not fault tolerant and there are many ways in which faults might disrupt the self-assembly process including, for instance, mechanical failure of the docking mechanism or failure of the power or communications busses across the docking mechanism. With real hardware operating over extended periods and multiple robots the probability of such faults is likely to be high. Thus planned work also includes extending the morphogenesis algorithm so that if faults are detected during self-assembly, the process modifies itself to compensate for those faults.

Acknowledgements The SYMBRION project is funded by the European Commission within the work programme Future and Emergent Technologies Proactive under grant agreement no. 216342.

References

1. Christensen, A., O'Grady, R., Dorigo, M.: Swarmorph-script: a language for arbitrary morphology generation in self-assembling robots. Swarm Intelligence **2**(2), 143–165 (2008). DOI 10.1007/s11721-008-0012-6
2. Doursat, R.: Organically Grown Architectures: Creating Decentralized, Autonomous Systems by Embryomorphic Engineering, *Understanding Complex Systems*, vol. 21, pp. 167–199. Springer Berlin / Heidelberg (2008). DOI 10.1007/978-3-540-77657-4_8
3. Gross, R., Dorigo, M.: Self-assembly at the macroscopic scale. Proceedings of the IEEE **96**(9), 1490–1508 (2008). DOI 10.1109/JPROC.2008.927352
4. Grushin, A., Reggia, J.A.: Automated design of distributed control rules for the self-assembly of prespecified artificial structures. Robot. Auton. Syst. **56**(4), 334–359 (2008). DOI http://dx.doi.org/10.1016/j.robot.2007.08.006

5. Guo, H., Meng, Y., Jin, Y.: A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network. Biosystems **98**(3), 193 – 203 (2009). DOI DOI: 10.1016/j.biosystems.2009.05.003
6. Kernbach, S., Meister, E., Scholz, O., Humza, R., Liedke, J., Ricotti, L., Jemai, J., Havlik, J., Liu, W.: Evolutionary robotics: The next-generation-platform for on-line and on-board artificial evolution. In: Proceedings of IEEE congress on Evolutionary Computation, pp. 1079–1086. Trondheim, Norway (2009)
7. Levi, P., Kernbach, S. (eds.): Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution. Springer (2010)
8. Liu, W., Winfield, A.: Implementation of an IR approach for autonomous docking in a self-configurable robotics system. In: T. Kyriacou, U. Nehmzow, C. Melhuish, M. Witkowski (eds.) Proceedings of Towards Autonomous Robotic Systems, pp. 251 – 258 (2009)
9. Murata, S., Kakomura, K., Kurokawa, H.: Toward a scalable modular robotic system. IEEE Robotics Automation Magazine **14**(4), 56–63 (2007). DOI 10.1109/M-RA.2007.908984
10. Nagpal, R.: Programmable self-assembly: Constructing global shape using biologically-inspired local interactions and origami mathematics. Ph.D. thesis, Massachusetts Institute of Technology (2001)
11. Rubenstein, M., Payne, K., Will, P., Shen, W.M.: Docking among independent and autonomous conro self-reconfigurable robots. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 3, pp. 2877–2882 (2004). DOI 10.1109/ROBOT.2004.1307497
12. Salemi, B., Moll, M., Shen, W.M.: SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system. In: Proceedings of Intenational Conference on Intelligent Robots and Systems, pp. 3636 – 3641. Beijing, China (2006)
13. Støy, K.: Using cellular automata and gradients to control self-reconfiguration. Robotics and Autonomous Systems **54**, 135–141 (2006)
14. Vaughan, R.: Massively multi-robot simulation in Stage. Swarm Intelligence **2**(2-4), 189–208 (2008). DOI 10.1007/s11721-008-0014-4
15. Werfel, J.: Biologically realistic primitives for engineered morphogenesis. In: the Seventh International Conference on Swarm Intelligence (ANTS2010), pp. xxx–xxx. Springer, Belgium (2010)
16. Yim, M., WHITE, P., PARK, M., SASTR, J.: Modular self-reconfigurable robots. Encyclopedia of Complexity and Systems Science (2009)
17. Yim, M., Zhang, Y., Roufas, K., Duff, D., Eldershaw, C.: Connecting and disconnecting for chain self-reconfiguration with polybot. IEEE/ASME Transactions on Mechatronics **7**(4), 442–451 (2002). DOI 10.1109/TMECH.2002.806221