robotica 2009
festival nacional de robótica

Parallel Task Execution, Morphology Control and Scalability in a Swarm of Self-Assembling Robots

Anders Lyhne Christensen Rehan O'Grady Marco Dorigo

***Abstract*— We investigate the scalability of a morphologically flexible self-assembling robotic system by measuring task execution performance. We use a scenario consisting of three subtasks — gap crossing, bridge traversal and object pushing. Each subtask can only be solved by a dedicated self-assembled morphology. To successfully complete the scenario, individual robots must autonomously assemble and disassemble to form morphologies appropriate to the subtask at hand. Environmental cues tell the robots when they have encountered a particular task. Parallel execution of tasks is possible when there is a sufficient number of robots. With simulated robots, we perform a series of experiments demonstrating the feasibility and the scalability of our system. We implement our distributed control using the scripting language SWARMORPH-script that has been used in previous studies to form morphologies with up to nine real robots.**

I. Introduction

Self-assembling robotic systems are composed of multiple autonomous agents that can physically connect to each other to form larger composite robotic entities. Two of the key potential benefits of self-assembling robotic systems are morphological flexibility and parallelism. Morphological flexibility is important because any robotic entity must have a morphology that is in some way appropriate to the task it needs to perform. In theory, the ability to form a wide range of different morphologies should allow future self-assembling systems to tackle a wider range of tasks than conventional monolithic robots. Such self-assembling systems may well comprise thousands or even millions of individual agents. In such large systems, parallelism will be the key to efficiency—different self-assembled robotic entities will be able to carry out different tasks at the same time. A well-designed self-assembling system should thus allow for massively parallel task execution.

In this study, we explore a scenario designed to investigate morphological flexibility and large scale parallelism. In our scenario, a series of subtasks must be completed. Each subtask is solvable by a dedicated self-assembled morphology, which is incapable of solving the other subtasks. The robots start at one end of the arena and perform phototaxis towards a light source at the other end of the arena. As they proceed, environmental cues indicate the presence of particular subtasks to be solved. When they encounter a subtask, the robots must assemble into the appropriate morphology for the subtask at hand. Once that subtask is complete, the robots disassemble and continue phototaxis. They are thus ready to assemble into another morphology as soon as they encounter another subtask. The nature of the subtasks allows for a degree of parallel execution.

Anders Lyhne Christensen (anders.christensen@iscte.pt) is with DCTI, Lisbon University Institute, Portugal. Rehan O'Grady (rogrady@ulb.ac.be) and Marco Dorigo (mdorigo@ulb.ac.be) are with IRIDIA, Université Libre de Bruxelles, Belgium.

In previous studies, we pioneered a distributed technique for morphology control in self-assembling systems [5], [17] using both real-robots and a dedicated simulation environment. We developed a scripting language with primitives that would allow robots to self-assemble into particular shapes and to disassemble [6], [16]. However, the sequence of morphologies formed was determined in advance by the experimenter, and the self-assembled entities did not carry out any tasks.

In this study, we extend our previous work to apply particular self-assembled morphologies to specific tasks. The self-assembled morphologies are now formed on demand in response to environmental cues. We demonstrate the feasibility of our enhanced system in a dedicated simulation environment. Using our scenario, we explore the behavior of our system under different configurations. We investigate the negative influence of interference by increasing the number of robots while keeping the size of the arena and the number of tasks constant. We investigate how the system scales by concurrently increasing the size of the arena, the number of robots and the number of tasks. The verisimilitude of the simulation environment was verified in a previous study [17].

The paper is organized as follows: In Sect. II, we discuss related work. In Sect. III, we present the *swarm-bots* robotic platform on which this study is based and describe our simulation environment. In Sect. IV, we present the three different tasks that the robots must accomplish through self-assembly and disassembly in our experiments. In Sect. V, we provide an overview of SWARMORPH-script. In Sect. VI, we present the results of our experiments. We discuss our results and conclude the paper in Sect. VII.

II. Related work

There is a large body of scientific literature on the distributed creation and control of robotic morphologies using inter-connectable components. The two principle approaches are self-reconfigurable systems and self-assembling systems. In self-reconfigurable systems [20], the components tend to be incapable of independent motion. In self-assembling systems [11], the components are themselves independent robots that can autonomously form physical connections with one another. In the latter case, the individual robots can be either externally propelled or self-propelled. Several different hardware architectures and control mechanisms have been proposed respectively for self-reconfigurable robotics [3], [14], [15], [19] and for self-assembling robotics [2], [7], [8], [10], [12].

The advantage of morphological flexibility is that it potentially allows a robotic system to carry out a wider range of tasks. Somewhat surprisingly, little work has

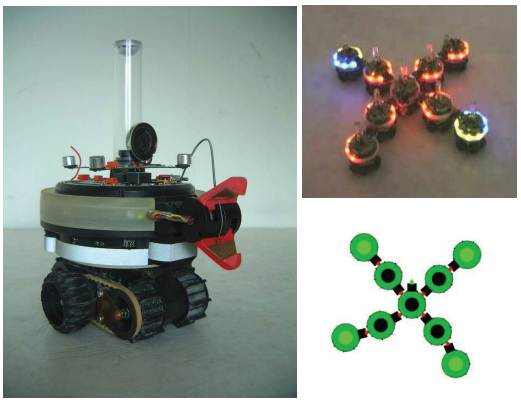

Fig. 1: Left: The *s-bot*. Top right: The star morphology formed with 9 real robots. Bottom right: The star morphology formed in our simulation environment.

directly focused on using self-reconfigurable systems to generate appropriate morphologies in response to task requirements [1] (and almost no work in the field of self-assembling systems). The advantage of self-assembling systems is that, as well as morphological flexibility, they offer the potential for parallel task execution. White *et al.* [18] used mathematical and simulation based models to analyze the scalability of their self-assembling system. However, in common with other works that consider scalable self-assembling systems with larger numbers of robots [13], the focus is on the ability of the system to self-assemble ever larger structures.

In this paper, our approach is different in that the robots form specific morphologies to solve different tasks, and that they carry out the tasks in parallel. We measure scalability, not by an internal measure of self-assembling efficiency, but rather by the external measure of task completion efficiency.

III. Robotics Platform

We conduct our experiments using a simulated version of the *swarm-bots* robotic platform. The platform consists of a number of mobile autonomous robots called *s-bots* (see Fig. 1) that are capable of forming physical connections with each other. Each *s-bot* is equipped with an XScale CPU running at 400 MHz, a number of sensors including an infrared ground sensors, proximity sensors, and light sensors. Physical connections between *s-bots* are established by a gripper-based connection mechanism. Each *s-bot* is surrounded by a semi-transparent ring that can be grasped by other *s-bots*. *S-bots* can advertise their location and/or internal state by means of eight sets of RGB-colored LEDs distributed around the inside of their semi-transparent ring.

The *s-bots* have an omni-directional camera that points upwards at a hemispherical mirror mounted above the *s-bot*'s turret in a transparent perspex tube. The camera records the panoramic images reflected in the mirror. Depending on light conditions, the camera can detect illuminated LEDs on other *s-bots* up to 50 cm away. The combination of the camera and the LEDs thus provides the *s-bots* with local, situated communication capabilities.

The experiments in this study were conducted in a simulation environment consisting of a specialized software simulator with a custom dynamics engine tailored to our robotic platform [4]. All the sensors and actuators that were used are simulated with reasonable accuracy by our simulation environment. We developed a control interface abstraction layer that allowed us to transfer our control programs between the simulator and the real robots without any modification. The control abstraction layer allowed us to run and test the same SWARMORPH-based control programs both in simulation and on real robots.

IV. Tasks and Morphologies

We have chosen three tasks: gap crossing, bridge traversal, and object pushing. None of these tasks can be solved by a single robot operating alone. Instead, the robots have to self-assemble and cooperate in order to accomplish each of the three tasks. Based on trial and error experimentation with real robots, we have designed the three tasks so that each task requires the robots to self-assemble into a dedicated morphology. Each morphology can solve one task and one task only, that is, the dedicated morphology that succeeds in solving one of the tasks will fail to solve if applied to either of the other two tasks. The tasks and their associated morphologies are shown in Fig. 2 and described in detail below.

A. The Gap Crossing Task

In this task, the robots must cross a 22 cm wide rectangular hole that runs the width of the arena. An *s-bot* can detect the gap based on readings from its infrared ground sensors. Of the *s-bot*'s four ground sensors, one points slightly forwards and one points slightly backwards. This allows an *s-bot* to detect a gap before falling into it. A gap of 22 cm was chosen because it is reliably passable by four real *s-bots* connected in linear morphology, while a three *s-bot* linear morphology will fail unless it is perfectly aligned (any smaller morphology always fails).

B. The Bridge Traversal Task

In this task, the robots must use a bridge to cross a 50 cm wide rectangular hole that runs the width of the arena. The bridge is made of two pipes spaced 17.5 cm apart, each with a diameter of 8 cm. The curvature of the pipes is sufficient that a moving *s-bot* cannot balance on a single pipe. The two pipes are also sufficiently far apart that the wheels of a single *s-bot* cannot make contact with both pipes at the same time. Thus, a single *s-bot* cannot traverse a bridge alone. However, a composite robotic entity comprised of two physically connected *s-bots* (appropriately oriented) can traverse a bridge, since it can make contact with both pipes at the same time—each *s-bot* touches one of the pipes. The curvature of the pipes does not cause the constituent *s-bots* of such an entity to topple, as the *s-bots* mutually support each other, see Fig. 2 (middle).

The on-board computer vision software does not enable the robots to estimate the width of a gap or to see the bridge. We have therefore placed a special reflective material before the bridged 50 cm gap to distinguish it from the 22 cm gap. The reflective material can be detected by an *s-bot* using its infrared ground sensors: readings

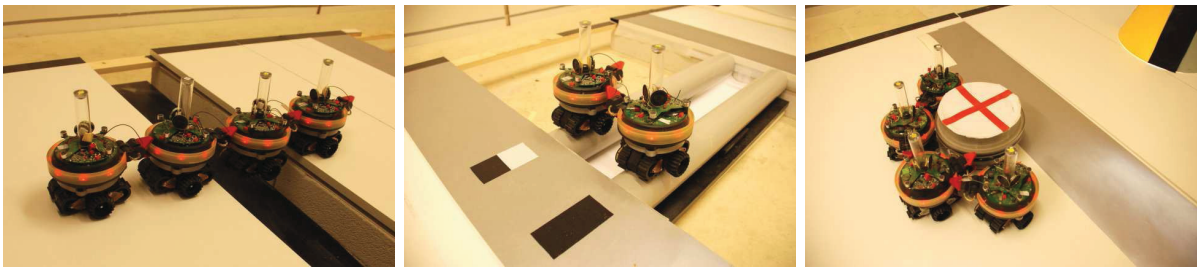

Fig. 2: The three tasks and the appropriate morphology for each tasks. Left: The gap crossing task (line morphology). Middle: The bridge traversal task (support morphology). Right: The object pushing task (shovel morphology).

are higher than for the normal arena floor. In order to determine the position of the bridge, we have put a distinct simple bar code in front of each pipe, see Fig. 2 (middle). The bar code is made up of different materials that can be detected by an *s-bot*'s ground sensors. Whenever a robot detects a bar code, it can use the bar code information to determine which pipe it is facing (left pipe or right pipe) and build the morphology to cross the bridge accordingly. We have also added reflective material on the far side of the bridge to allow the robots to detect when they have successfully crossed the bridge.

C. The Object Pushing Task

In this task, the robots have to perform cooperative transport by pushing two or more objects 30 cm towards the light source. The objects have a dimension and weight that prevents a single *s-bot* from pushing them. In fact, a shovel shape formed by four robots is necessary to reliably shift an object, see Fig. 2 (right). We use objects with a diameter of 20 cm positioned in front of a 30 cm expanse of reflective material. The robots are programmed so that when they have reached the end of the reflective material, they disassemble and move back across the reflective material to search for more objects. The objects are wrapped in the same reflective material. An object that should be shifted can thus be detected by an *s-bot* based on proximity sensor readings—because of the reflective material, the readings for the object are higher than those for either other *s-bots* or for walls.

V. Methodology

We have developed a distributed control scheme that allows *s-bots* to respond to the obstacles described in Sect. IV and to self-assemble into specific morphologies[1]. Each *s-bot* is autonomous and only local, situated, color-based communication is used between the *s-bots*. Whenever an *s-bot* detects the presence of a task that requires a larger robotic entity to be self-assembled, it starts a new self-assembly process by illuminating its LEDs in a particular color configuration. The color configuration indicates a point on the *s-bot*'s body where another non-attached *s-bot* should grip and a corresponding orientation which the gripping *s-bot* should assume. We term such a color configuration a *connection slot* [5].

When an *s-bot* has gripped another *s-bot*, the two *s-bots* initiate communication by changing the color configuration of their LEDs. The communication system allows for the transmission of strings. Through this communication, the newly connected *s-bot* receives instructions on how to extend the local structure. Following these instructions, the newly connected *s-bot* in turn attracts other *s-bots* by opening a new connection slot itself. When a subsequent new *s-bot* attaches, it once again initiates communication, and is told in turn how to extend the structure. As this process repeats itself, the morphology grows accordingly.

A. The SWARMORPH-Script Language

We abstracted basic behaviors such as *phototaxis*, *invite connection*, *send rule ID*, and *disconnect*, into a set of control primitives. We used these control primitives to build a morphology creation language (SWARMORPH-script) that can be executed on real *s-bots* [6]. The language allows for explicit high-level expression of distributed rules for morphology growth. Below, we provide a summary of some of the primitives available in SWARMORPH-script:

- `Phototaxis`: Perform phototaxis until an obstacle has been encountered or overcome.
- `OpenConnSlot`: Invite a connection at a certain location.
- `Connect`: Find and connect to an *s-bot* inviting a connection.
- `SendRuleID`: Send the ID of a rule.
- `ReceiveRuleID`: Receive the ID of a rule.
- `Notify`: Notify a physically connected *s-bot*.
- `Disconnect`: Open the gripper to disconnect from the morphology.
- `Retreat`: Retreat for a certain amount of time.
- `if, then, end`: Branch based on the type of obstacle encountered or based on the rule ID received.

B. The Script

In this section, we describe the script that is used to solve our three task scenario. We describe the overall functioning of the script, and for illustrative purposes present a section of the script, see Script 1. We show the global structure of the script, and focus on the part of the script that builds the

[1]Note that the sensory equipment available on the *s-bot* platform is not sufficiently sophisticated to allow for a truly adaptive morphological response mechanism. Instead, as discussed in Sect. IV, we place cues in the environment that are detectable by the s-bots. The cues uniquely identify the different tasks, and trigger the formation of the appropriate morphology.

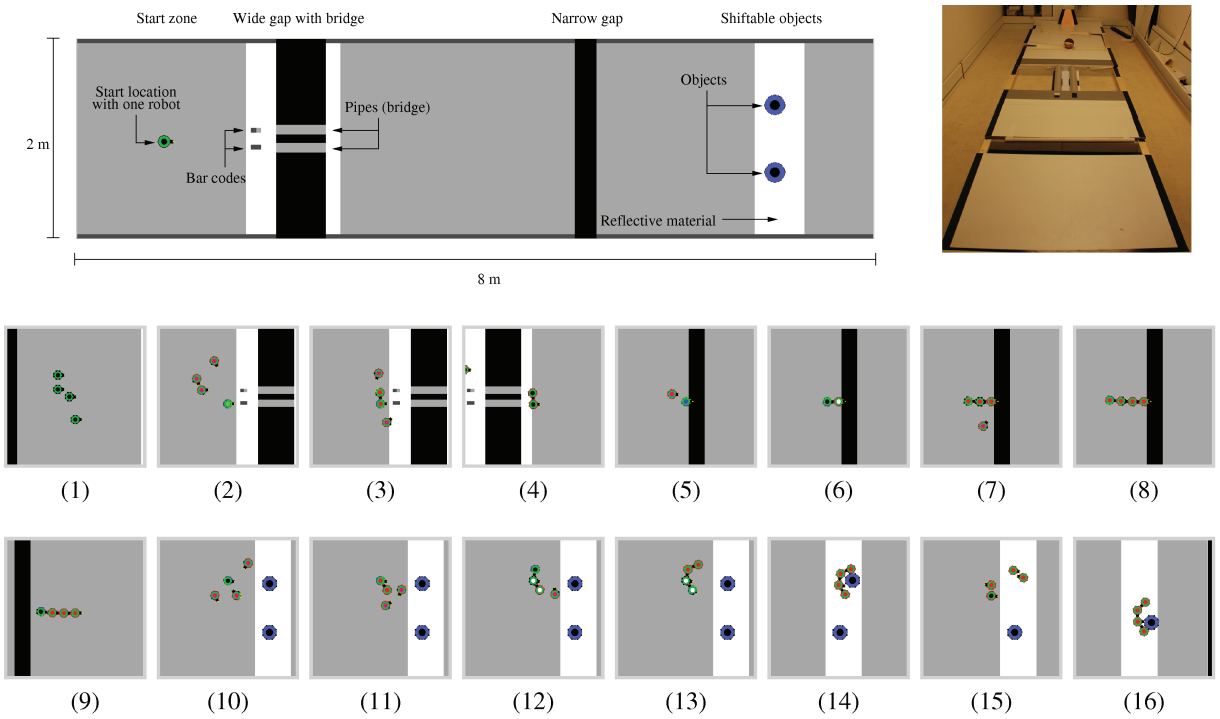

Fig. 3: Top left: A simulated arena (8 m x 2 m). Top right: The real arena (5.0 m x 1.8 m). Bottom rows: Four *s-bots* completing the scenario by first crossing the bridge, then the narrow gap, and finally shifting the two objects. In simulation, the light source is placed on the far right of the arena (not shown). For debugging purposes, the centers of the *s-bots* visually indicate the current controller state of the *s-bots*.

two-*s-bot* morphology necessary to cross the bridge. Due to space considerations we do not show the full script. The full script can be found in the accompanying online material:

`iridia.ulb.ac.be/supp/IridiaSupp2009-002/index.html`

All *s-bots* execute this SWARMORPH-script. Initially, the *s-bots* perform phototaxis individually. When one of the *s-bots* encounters an obstacle, the *s-bot* illuminates its LEDs in order to invite another *s-bot* to connect (it opens a connection slot). When an *s-bot* performing phototaxis sees that another *s-bot* is inviting a connection, it ceases to perform phototaxis and instead tries to physically connect to the inviting *s-bot*. When a successful connection has been formed, communication is initiated.

Fig. 3 shows an example of an arena used in simulation, the real arena, and an example run in which the script presented above is executed on four *s-bots*. A light source is placed on the far right of the arena (not shown), grey indicates normal arena floor, black indicates a hole, while white indicates reflective material. In Fig. 3(1), the four *s-bots* start by performing phototaxis. One of the *s-bots* detects a bar code indicating the presence of a bridge; the *s-bot* retreats off the reflective material and opens a connection slot (Fig. 3(2)). The other *s-bots* detect the invitation to connect and one of them manages to attach (Fig. 3(3)). The *s-bot* that detected the bar code communicates with the newly attached *s-bot* and instructs it to start crossing the bridge. When the two *s-bots* have crossed the bridge (Fig. 3(4)), they disassemble and continue performing phototaxis. One of the two *s-bots* to first cross the bridge then detects the narrow gap and initiates the formation of a line morphology. The other two *s-bots* have also crossed the bridge and contribute to the line morphology (Fig. 3(5-8)). After crossing the gap (Fig. 3(9)), the *s-bots* disassemble and continue moving towards the light source. One of the *s-bots* detects an object (Fig. 3(10)) and starts the formation of a shovel morphology (Fig. 3(11-13)). When assembled into the shovel morphology, the *s-bots* then push the object across the reflective material (Fig. 3(14)), disassemble and retreat back across the reflective material (Fig. 3(15)). The other object is encountered, a morphology is assembled, and the other object is pushed (Fig. 3(16)).

VI. Experiments and Results

In this section, we describe a series of experiments that we carried out in different arenas. We first test our script in a basic task execution experiment, where four robots carry out the three subtasks in sequence. We then test the negative influence of interference in our system, by increasing the number of robots while keeping the arena configuration constant. Finally, we test the scalability of our system by creating a series of progressively larger arenas that allow the tasks to be carried out in parallel, and conduct experiments with correspondingly larger numbers of robots.

A. Basic Task Execution

We conducted 100 experiments with 4 *s-bots* in a 8 m x 2 m arena containing a bridged gap, a narrow gap, and two pushable objects. We used 4 *s-bots*, as this is the minimum number of robots able to complete the scenario

Script 1: Solve three subtasks in an unknown order.

```
Label: "PhototaxisAndLookForTasks"
Phototaxis();
if right-bridge-cue-detected then
    # Retreat off the reflective material
    Retreat();
    # Invite new connection from the left
    OpenConnSlot(left);
    # Send instructions to the connected s-bot
    SendRuleID(1);
    # Cross bridge
    Phototaxis();
    # Restart script
    Jump(PhototaxisAndLookForTasks);
end
else if left-bridge-cue-detected then
    # Retreat off the reflective material
    Retreat();
    # Invite new connection from the left
    OpenConnSlot(right);
    # Rest of the code is identical to code above
    ...
end
else if hole-detected then
    # Start a line morphology
    ...
end
else if object-detected then
    # Start a shovel morphology
    ...
end
else if conn-slot-detected then
    # Connect to a connection slot
    Connect();
    # Receive instructions
    ReceiveRuleID();
    # If a bridge is ahead
    if receivedruleid = 1 then
        # Phototaxis across the bridge
        Phototaxis();
        # Disconnect from the seed
        Disconnect();
        # Restart script
        Jump(PhototaxisAndLookForTasks);
    end
    # Logic for the other morphologies
    ...
end
```

— four *s-bots* are needed both to cross the narrow gap (line morphology) and to push the object (shovel morphology). In each experiment, we recorded the time it took the four *s-bots* to navigate through the arena and to push both of the two objects 30 cm.

At the start of each experiment, the *s-bots* were placed in the starting zone and oriented to face the light source. We let each experiment run for 6,000 simulated seconds (= 100 minutes). The results are summarized in Tab. I. In ninety-four of the experiments, the four *s-bots* succeeded in navigating the arena and pushing both of the objects the required distance.

We witnessed two types of failure that prevented one or both of the objects from being pushed in six of the experiments. Firstly, there are sometimes 'robot casualties' during task execution. We consider a robot to be a casualty if it falls into one of the gaps. When one or more robots fall into a gap before both objects have been pushed the requisite distance, there are then insufficient remaining *s-bots* to complete the scenario. Secondly, if the *s-bots* form

TABLE I: Results summary of experiment with 4 *s-bots* in an 8 m x 2 m arena and two objects to push.

0 objects pushed	1 experiments
1 object was pushed	5 experiments
2 objects were pushed	94 experiments
Average time, 1st object	1,150 s (st.dev 310 s)
Average time, 2nd object	1,803 s (st.dev 332 s)

a slightly misaligned shovel morphology, the object can slide off the side of the shovel before it has been shifted 30 cm. As a result, the object remains in the center of the reflective band and can no longer be detected by the *s-bots*.

In one experiment, neither of the two objects were successfully pushed. This occurred due to a misaligned shovel morphology in both cases. In another five experiments, only one of the two objects was pushed (see Tab. I). One of these experiments failed due to robot casualties, and four of these experiments failed due to misaligned morphology growth.

B. Negative Influence of Interference

Both types of failure that we saw in the previous section are caused by interference (for more details on interference and its potential role in controller design, see [9]). Interference occurs when a high local density of robots results in collisions (although the robots perform obstacle avoidance, this mechanism is overwhelmed when the density is sufficiently high). Collisions lead to robot casualties when one of the colliding robots is pushed into a gap. Collisions lead to misalignment when a robot that is inviting a connection is displaced or rotated by a collision with another *s-bot*.

To determine the influence of interference on task completion performance, we ran an additional set of experiments with a varying number of *s-bots* in the same 8 m x 2 m arena that we used in Subsect. VI-A (see Fig. 3(top left)). In each experiment, the *s-bots* were initially placed in the starting zone and oriented to face the light source.

The results are summarized in Fig. 4. For each experimental setup with a given number of *s-bots*, we performed 100 replications. In each replication, we varied the initial placements and initial seed for the random number generator. The results for each set of experiments are summarized by two bars. The wide bars indicate the average task completion time and standard deviation observed in 100 replications of the experimental setup. The narrow bars denote the percentage of robot casualties.

As the results indicate, the average performance initially increases as more *s-bots* are added. However, at a group size of 18 *s-bots*, the average performance begins to decrease. Furthermore, the percentage of robot casualties (the narrow bars in Fig. 4) increases monotonically with the robot density. In the four *s-bots* experiments, we observed one robot casualty in a single experiment, yielding a robot casualty percentage of 0.25%. When 30 *s-bots* are present in the same arena, the robot casualty percentages is 20.90% ($\approx$ 6 *s-bots*/experiment on average).

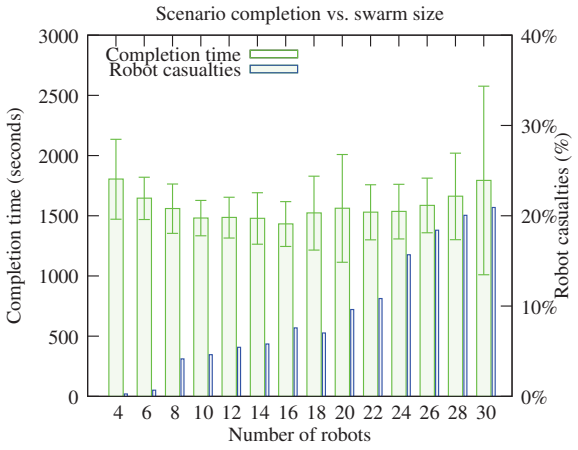

Fig. 4: Scenario completion times and robot casualties for swarms of different sizes in an 8 m x 2 m arena with pushable objects.

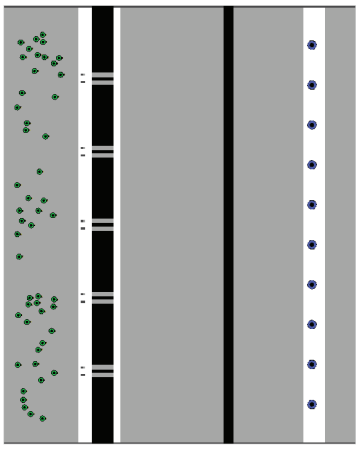

Fig. 5: An example of an arena for scalability experiments with 50 *s-bots*. The width w of the arena is 10m, the number of bridges b is 5, and the number of pushable objects o is 10.

C. Scalability

In order to evaluate the scalability of our approach, we ran a series of experiments with progressively larger numbers of *s-bots* and correspondingly larger arenas. We varied the size of the initial robot population from 100 robots to 1,000 robots in increments of 100. We performed 100 replications for each population size. Each experiment was run for 1,500 simulated seconds (= 25 minutes).

For each population size, we set the width of the arena, the number of bridges and the number of objects as a function of the number of robots in the population. If n is the number of *s-bots* in a given experimental setup, the arena is 8 m long, $w = n/5$ meters wide, contains $b = n/10$ bridges, and $o = n/5$ pushable objects. The bridges and pushable objects are uniformly distributed along two lines running the width of the arena. Note that to obtain the arena that was used in the experiments of the previous two sections, we would need an initial population size of 10 robots ($n = 10$). An example of an arena corresponding to an initial robot population of 50 *s-bots* ($n = 50$) is shown in Fig. 5.

The results are summarized in Fig. 6. Each bar denotes the average number of objects pushed by a swarm of a fixed size over the 100 replications of the experiment. Each bar is annotated with the standard deviation for the result set.

In Fig. 6, we have added a least squares fit line ($y = 0.117 \cdot x$). As the results show, the task execution performance scales linearly with the number of *s-bots*. Linear scalability should not come as a surprise: the control is completely decentralized and each *s-bot* acts based only on what it senses in its immediate vicinity. We therefore expect that the linear scalability trend would continue beyond swarms of 1,000 *s-bots* (however, given the computational resources required, we have been unable to confirm this).

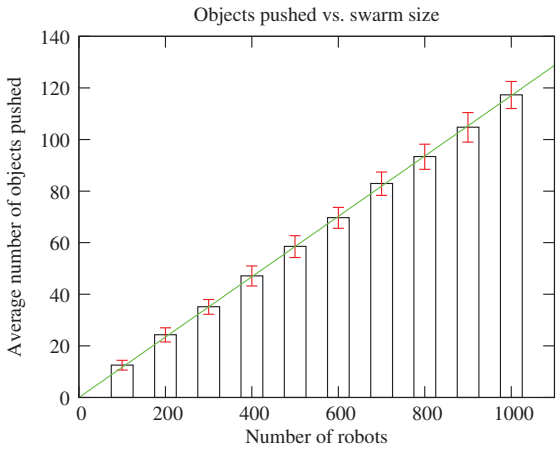

Fig. 6: Scalability

VII. Conclusions

We have presented a scenario in which robots have to solve three different tasks. In order to solve the different tasks, the robots have to cooperate by self-assembling into specific morphologies appropriate to each task.

We conducted extensive simulation-based experiments to investigate issues related to interference between robots. We found that high robot densities resulted in a lower performance and an increase in robot casualties. In another set of experiments, we investigated scalability by increasing the number of robots and tasks in the scenario. We found that the task execution performance scales linearly with the number of robots and number of tasks — at least up to 1,000 robots. Given our decentralized control approach, we expect this trend to continue for even larger swarms.

We are currently conducting experiments on real robotic hardware using the same SWARMORPH-script based control program that we have used in the simulation-based experiments presented in this study. Our ongoing research concerns the cooperation between meta-entities, that is, cooperation between two or more self-assembled robotic entities.

Acknowledgements

This work would not have been possible without the innovative robotic hardware developed by Francesco Mondada's group at the Laboratoire de Systeme Robotiques

(LSRO) of EPFL. This work was supported by the *SWARMANOID* project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888 and by the *VIRTUAL SWARMANOID* project funded by the F.R.S.-FNRS. Marco Dorigo acknowledges support from the F.R.S.-FNRS, of which he is a Research Director.

REFERENCES

[1] H. Bojinov, A. Casal, and T. Hogg. Emergent structures in modular self-reconfigurable robots. In *IEEE International Conference on Robotics and Automation (ICRA 2000)*, pages 1734–1741. IEEE Computer Society Press, Los Alamitos, CA, 2000.

[2] H. B. Brown, J. M. V. Weghe, C. A. Bererton, and P. K. Khasla. Millibot trains for enhanced mobility. *IEEE/ASME Transactions on Mechatronics*, 7(4):452–461, 2002.

[3] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *International Journal of Robotics Research*, 23(9):919–937, 2004.

[4] A. L. Christensen. Efficient neuro-evolution of hole-avoidance and phototaxis for a swarm-bot. Technical Report TR/IRIDIA/2005-14, IRIDIA, Université Libre de Bruxelles, Belgium, 2005. DEA Thesis.

[5] A. L. Christensen, R. O'Grady, and M. Dorigo. Morphology control in a multirobot system. *IEEE Robotics & Automation Magazine*, 14(4):18–25, 2007.

[6] A. L. Christensen, R. O'Grady, and M. Dorigo. SWARMORPH-script: A language for arbitrary morphology generation in self-assembling robots. *Swarm Intelligence*, 2(2-4):143–165, 2008.

[7] R. Damoto, A. Kawakami, and S. Hirose. Study of super-mechano colony: concept and basic experimental set-up. *Advanced Robotics*, 15(4):391–408, 2001.

[8] T. Fukuda, M. Buss, H. Hosokai, and Y. Kawauchi. Cell structured robotic system CEBOT: Control, planning and communication methods. *Robotics and Autonomous Systems*, 7(2-3):239–248, 1991.

[9] D. Goldberg and M. J. Matarić. Interference as a tool for designing and evaluating multi-robot controllers. In *Proceedings of the National Conference on Artificial Intelligence*, pages 637–642. John Wiley & Sons, Hoboken, NJ, 1997.

[10] R. Groß, M. Bonani, F. Mondada, and M. Dorigo. Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, 22(6):1115–1130, 2006.

[11] R. Groß and M. Dorigo. Self-assembly at the macroscopic scale. *Proceedings of the IEEE*, 96(9):1490–1508, 2008.

[12] S. Hirose, T. Shirasu, and E. F. Fukushima. Proposal for cooperative robot "Gunryu" composed of autonomous segments. *Robots and Autonomous Systems*, 17:107–118, 1996.

[13] K. Hosokawa, I. Shimoyama, and H. Miura. Dynamics of self-assembling systems: analogy with chemical kinetics. *Artificial Life*, 1(4):413–427, 1994.

[14] E. Klavins, R. Ghrist, and D. Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, 51(6):949–962, 2006.

[15] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-tran: Self-reconfigurable modular robotic system. *IEEE-ASME Transactions on Mechatronics*, 7(4):431–441, 2002.

[16] R. O'Grady, A. L. Christensen, and M. Dorigo. Autonomous reconfiguration in a self-assembling multi-robot system. In *Ant Colony Optimization and Swarm Intelligence, Sixth International Conference, ANTS 2008*, pages 259–266. Springer-Verlag, Berlin, Germany, 2008.

[17] R. O'Grady, A. L. Christensen, and M. Dorigo. SWARMORPH: Multi-robot morphogenesis using directional self-assembly. *IEEE Transactions on Robots*, 2009. In press.

[18] P. J. White, K. Kopanski, and H. Lipson. Stochastic self-reconfigurable cellular robotics. In *Proc. of the 2004 IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 2888–2893. IEEE Computer Society Press, Los Alamitos, CA, 2004.

[19] M. Yim, K. Roufas, D. Duff, Y. Zhang, C. Eldershaw, and S. B. Homans. Modular reconfigurable robots in space applications. *Autonomous Robots*, 14(2-3):225–237, 2003.

[20] M. Yim, W. M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.