Assessing the Effect of Self-Assembly Ports in Evolutionary Swarm Robotics

Kazi Shah Nawaz Ripon*, Eirik Jakobsen*, Christopher Tannum*, Jean-Marc Montanier†
*Department of Computer and Information Science, Norwegian University of Science and Technology, Norway
†Protolab – Softbank Robotics Europe
Email: ksripon@idi.ntnu.no, eirikjak@gmail.com, christannum@gmail.com, montanier.jeanmarc@gmail.com

***Abstract*—Self-assembly in swarm robotics is essential for a group of robots in achieving a common goal that is not possible to achieve by a single robot. Self-assembly also provides several advantages to swarm robotics. Some of these include versatility, scalability, re-configurability, cost-effectiveness, extended reliability, and capability for emergent phenomena. This work investigates the effect of self-assembly in evolutionary swarm robotics. Because of the lack of research literature within this paradigm, there are few comparisons of the different implementations of self-assembly mechanisms. This paper reports the influence of connection port configuration on evolutionary self-assembling swarm robots. The port configuration consists of the number and the relative positioning of the connection ports on each of the robot. Experimental results suggest that configuration of the connection ports can significantly impact the emergence of self-assembly in evolutionary swarm robotics.**

***Keywords*—Evolutionary robotics, self-assembly, machine learning, connection port configuration, docking mechanism.**

I. Introduction

By drawing inspiration from social insects [1], [2] and other self-organizing systems [3], [4], swarm robotics approaches the coordination of a number of autonomous robots, which need to interact and to cooperate to achieve a common goal [5]. The core idea is to capitalize on simple interactions among robots in order to solve complex problems by means of emergent behaviour.

Among these behaviours, self-assembly is the autonomous organization of components into patterns or structures [6]. It can provide multiple advantages in robotics such as robustness through redundancy and cost reduction through design of large number of robots [7]. Another crucial benefit is the versatility achieved when robots self-assemble into new structures based on the specific task to solve [8]. For example, the movement of robots is improved when they are able to overcome larger obstacles in an environment. In order to foster most of this advantage, it is crucial to learn when and how to self-assemble to face the environmental conditions at hand. This is the objective of functional self-assembly [9].

In order to achieve functional self-assembly, robots have to learn autonomously the most suited behaviours. To date, autonomous learning of self-assembly has been achieved through reinforcement learning [10] and evolutionary algorithms [11]. This work will consider only the evolutionary algorithms. Evolutionary Robotics (ER) studies how to automate the design of control systems for autonomous robots, using algorithms based on Darwinian evolution [9]. This approach has been already successfully used in the design of swarm behaviours [12], [13]. Previous studies on self-assembly have shown the difficulty to evolve such behaviour [14]. A key goal of this work is to study the mechanisms that can promote the learning of functional self-assembly through evolution.

Previous studies on the evolution of self-organization have focused on using differing self-assembly mechanisms, such as the architecture used by the self-assembly system, the actions that are performed by the robots to achieve an organized structure, or the docking mechanism hardware [9], [14], [15]. However, existing works do not provide any direct comparison among the mechanisms in order to justify which is the most suitable to promote the evolution of self-assembly. This hinders further research on the topic. This work studies one specific aspect of the self-assembly mechanism: the configuration (number and relative positioning) of the connection ports. Mainly, we studied the influence of ports configuration on the emergence of self-assembly when the robots are given basic learning capabilities.

For the experiment, we designed a simple predator/prey scenario, where the evolved robots (prey) can self-assemble to gain certain advantages over its predators. The experiments are simulated with a heavily modified version of *roborobo* platform [16].

The remainder of this paper is organized as follows. Section II presents the state of the art. Section III contains the system description and implementation, including modifications which were made to the existing roborobo framework. Section IV describes the results of the experiment, as well as an analysis and reasoning for their given state. Section V concludes the work including scopes for future work.

II. Related Works

Multiple works have discussed and compare the various self-reconfigurable robots [17]–[20]. Unfortunately, most of them introduce new platforms without comparing their performances to previous works [17]–[19]. To date, there exists no such work justifying a self-assembly system as better than others considering any specific property.

Latest articles for the design of self-reconfigurable robots propose versatile robots able to emulate previously cited robots [21], [22]. Nevertheless, their performance with regards to the simulated counter-parts is not studied. [20] compares

978-1-5090-4240-1/16/$31.00 ©2016 IEEE

various self-reconfigurable robots without the purpose of introducing a new robot. The comparison is done based on objective and measurable criteria such as number of degrees of freedom. However, the properties of these robots are not studied with relation to the performance of the robots towards specific tasks. For example, the number of docking ports with regards to the speed of reconfiguration is not studied. Moreover, as in other articles cited, the challenges envisioned do not take into account the possibility to learn autonomous behaviours on robots. Therefore, the ease to learn when and how to self-assemble is not taken into account.

Our goal is to study the influence of properties of self-assembly mechanisms with regards to the ease with which robots can learn to use them. In this article we will analyze the influence of the number of docking ports as well as their positioning.

III. Methodology

Experiments were implemented on an adapted version of the *roborobo* simulator [16], enabling the self-assembly of robots. This section includes the motivation for our experimentation, the experimental environment, description of roborobo and the modifications that were made to the existing software.

A. Experiment Motivation

Swarm behaviour found in nature displays how simple agents can self-assemble for various tasks such as defending against predators [23]. Our experiment takes inspiration from these behaviours and involves two groups of robots: predator and prey. In our experiment, there are nutrients which the preys can feed on to replenish their energy. The predators are larger than the preys and intend to feed on them. The preys can protect themselves from a predator by forming a larger structure which the predator is unable to eat. Moreover, when the preys are assembled, they may eat the larger predator to replenish energy as well. To gain an advantage the preys must form a large enough structure to prevent them from being consumed by the predators. Maintaining the assembled structure costs energy, which gives the preys an incentive to disassemble once the structure is no longer required. Therefore, in order to achieve the best result possible, i.e. the highest rate of survival, the preys should evolve an efficient functional self-assembly.

B. Experimental Environment

The environment is a rectangular box (Fig. 1), containing the robots (preys and predators) and nutrients. Initially, all the positions are random. When a nutrient is consumed, a new one is placed at a random position in the environment. The preys are endowed with a local communication module that can be used to send an array of floating point numbers to its neighbours in the self-assembly structure. The influence of the communication module and the values that the communication packets take are governed by a neural network, whose weights are evolved. The focus of our work is to analyze how the properties of the docking ports (number and angles between ports) will influence the capacity for the preys to self-assemble.

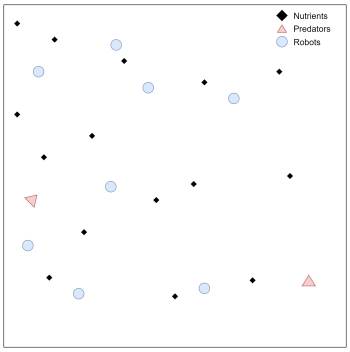


Fig. 1: Initial configuration of the environment.

C. Roborobo Overview

Roborobo is a light-weight multi-platform simulator for extensive robotics experiments, based on e-puck [24] like robots: two wheels, distance sensors, local communication module. Its programming interface is divided into three components (*World observer, Agent observers, Agent controllers*). Each robot in the simulation receives an instance of the agent controller and the agent observer. The roborobo framework enables programmers to implement the robot behaviour in the agent controller, and use the agent observer and world observer to access states about the given agent and the world. Each robot in the simulation also receives a component called the *world model* which contains an agent specific representation of the current state of the outside world. This model also includes states such as sensors readings and current velocity. In each simulation step, the observers are run before any of the agent controllers are run to have a stable snapshot of the world between each update.

D. Roborobo Modifications

Roborobo does not support self-assembly out of the box. Self-assembly support, therefore, requires adding additional abstractions to roborobo. This section includes the different modifications that were made to the roborobo framework to support self-assembly. The modified version of roborobo that we used in this work and the system configuration can be found at *https://github.com/christjt/ntnu-project-2016*.

1) Robot Groups: The high-level abstraction that encapsulates the behaviour for connected robots is called a robot group. All robots that are capable of self-assembly are robot groups. Single robots that are not connected are simply robot groups with one member. The responsibilities of the robot group are to take care of group movement, handling collisions, connecting, and disconnecting.

a) Movement: The movement of a group is decided by first converting the desired direction and velocity into a translation vector. The combined movement of the group is then decided by averaging the translation vectors from each member in the group. Once the translation vector for the group is computed, it can be applied to each member of the group by converting it back to the format of direction and velocity that roborobo uses.

Note that, in accordance with the philosophy of roborobo (simplified 2d simulation), our aim is to compute a simplified movement for the groups of robots. As a consequence, these movements do not take into account the alignment of the wheels of each robot, which in a more complex simulation would slow down the group of robots. Our simulation also discard any effect due to the position of the docking ports around the robots.

b) Collisions: Roborobo already performs collision detection for robots, but some additional logic is required in robot groups. The collision behaviour for robots is that if they collide with a solid object, they stop. This behaviour is a problem for groups consisting of more than one robot; because if one robot collides, it may get left behind by the rest of the group. This issue is solved by backing up the position of each robot in a group before applying the computed translation. If a robot in the group collides with something, then the robots in the group are reverted to their original position.

c) Connections: The connections between robots in a group are considered as a graph, where the robots are nodes and connections are edges. Connecting robots can then be treated as simply adding an edge between the two nodes representing the robots. Similar to connecting, disconnecting consists of removing the edge between the nodes representing the robots. In the case that a robot has multiple connections in the group, extra care has to be taken. This is because, removing one edge may split the graph into two smaller sub-graphs. If this occurs, the two sub-graphs are treated as two new separate groups. It can be determined if a robot can simply be disconnected, or if we have to split the group by finding the existence of a cycle that leads back to the disconnecting robot. The existence of a cycle is determined by first removing the edge representing the connection, and then performing a depth first search.

2) Self-assembly Mechanism: As mentioned in Section I, in this work, we will explore one specific property of the self-assembly mechanisms: the number of connection ports. Within our simulation, the robots can attempt a connection at any time. This requires a procedure to make sure the attempted connections are valid. Two requirements have to be met before a connection can be established between robots. First, the ports have to be spatially sound. Here, the distance between the ports has to be less than a threshold value. Next, the position of the ports has to be geometrically sound. Here, the angle between the connection ports has to fit within a threshold range.

3) Local Communication: The robots are equipped with a communication module that allows local communication with their connected neighbours. The message format is very simple – the robots can only broadcast message packets of floating point numbers. The size of the packets is equal to the number of connection ports. At the receiving end, the packets sent by neighbours are aggregated into a single packet. The aggregation is performed by adding the components of each message packet with the corresponding components from the other message packets. In the robot controller, messages from the communication module are fed into dedicated message input neurons, and the value of the message output neurons are broadcast to the neighbours. Additional experimentation has shown that communication modules are necessary for the synchronisation of robots so as to form a self-assembled group. The importance of communication modules is not within the scope of this article and therefore not presented.

4) Predators: An important part of the simulation environment is the predator robots. These robots are not under the influence of the EA, rather uses a simple pre-programmed controller. The predator robots have two basic actions that can affect the system. They can either eat prey or explore the environment.

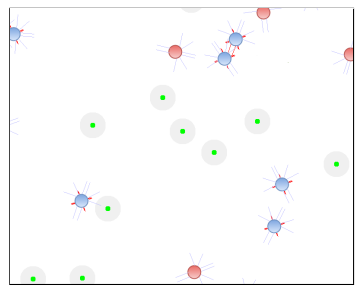


Fig. 2: Screenshot of a simulation. The predators are marked as red and the robots guided by the EA are marked as blue.

As seen in Fig. 2, the predators use six sensors to decide if they are colliding with an object and what that object is. Concerning exploration, the predator controller uses a simple object avoidance strategy. If the predator robots sense an object on any of its side sensors, it will change its rotational velocity to avoid this object. A slight random rotational velocity is added such that its movement becomes more dynamic in the case where there is no collision. This object avoidance behaviour is always employed as long as it does not sense a single robot (prey not self-assembled). In this case, it will have the opposite behaviour and turn towards the object, as it intends to consume the prey.

In addition to movement, the predators can also consume a robot when there is a collision between a prey robot and a predator robot. If a robot is consumed, it is removed from the environment and its lifetime is recorded to use in the fitness function of the EA. For a predator to be able to eat a robot, the robot must not be part of a robot group of a size less than some configurable threshold. The default threshold is set to two for the simulations run in this experiment. The reason for this is to give the regular robots some advantage of self-assembling in hopes of potentially evolving this behaviour.

5) Energy Drain: In addition to the environmental hazards of predators, the robots also face the issue with having a limited energy supply. If a robot loses all of its energy, then it will die and be removed from the environment. We added an energy hazard for the robots to give them a non-trivial self-assembling scenario. If predators were the only environmental obstacle, then the robots would most likely develop a self-assembly strategy and hold this construction for the remaining period of the simulation. It is intended that the

robots would have an environment which allowed strategies such as dissembling the self-assembly structure to evolve or perhaps not even self-assemble at all. This reasoning is also the justification for giving the robots additional energy drain when assembled.

Additional experiments not presented in this paper have shown the validity of this approach. The absence of energy drain leads to the evolution of a self-assembly strategy holding for all the simulation.

E. Artificial Neural Network (ANN)

In the robot controller, the neural network is responsible for making decisions. For this work, an ANN that has gained popularity in regards to robotics known as Continuous-Time Recurrent Neural Network (CTRNN) [25] was implemented. The information that the network is provided with is sensor readings, the connection status of each docking port, messages from the communication module, and finally the current energy level of the robot. The number of hidden layers and the number of neurons in each of them is configurable at run-time. The robots have to be able to detect, and distinguish between predators, other robots, food, and the environment. For this, the robots are equipped with six sensors. The topology uses 24 nodes to represent the inputs where there is one node per sensor for each of the four different objects that can be detected. With this configuration, the robot can distinguish between multiple types of objects at the same time.

F. Evolutionary Algorithm (EA)

In this work, a random initialization of genotype was implemented. The genotype contains weights, gains and time constants for the neural network used in the robot controller. During run-time, the genotype size is fixed and depends on the size of the neural network. The structure of the genotype is an array of double precision floating point numbers in the range $[-1, 1]$. Each number in the genotype represents a specific weight, gain, or time constant in the neural network.

For mutation, an incremental mutation operator was implemented, which re-rolls the selected double (x) to a new double ($x \in [-1.0, 1.0]$), where a weight is chosen for mutation at some percentage. However, instead of completely re-rolling its value, it only differs from its original value by a certain threshold to provide smoother results. For selection, we used binary tournament selection. At the end of each generation, the best 2 fittest individuals are allowed entry into the next parent selection unchanged to implement elitism.

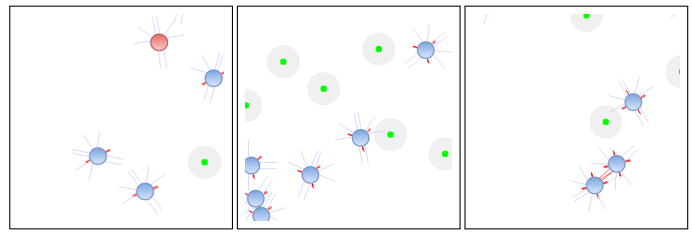
The lifetime of a robot was used as the fitness function. The fitness score, $f(G)$, of some genome (G) can be calculated as:

$$f(G) = \frac{\sum_{i=1}^{n} G(L_i)}{nL_{max}} \tag{1}$$

where, L_i represents the lifetime of robot i during simulation, n is the total number of robots, and L_{max} is the maximum lifetime of a robot.

IV. Results and Discussion

The simulations were done with different number of connection ports. The goal of running these simulations is to make correlations between its results to show how configuring the hardware mechanism can affect a self-assembly system. Snapshots of simulations using different number of ports can be viewed in Fig. 3.



(a) 2-ports (b) 3-ports (c) 4-ports

Fig. 3: Three different port configurations used in simulation.

Statistical significance is tested thanks to the Mann-Whitney Rank Test (as implemented in *scipy* version *0.13.3*) applied to the distributions obtained at the last generation of the runs. Results are considered as statistically different if the $p-value$ is bellow 0.01. Common configuration parameters for the simulations are as follows:

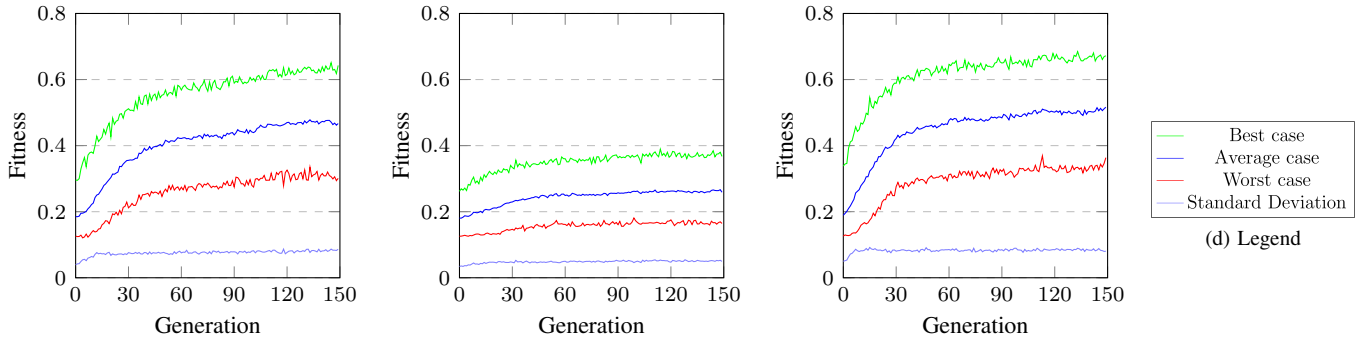
Parameter	Value
Number of Robots	20
Iterations per Generation	10000
Scenarios	3
Generations	150
Mutation Rate	0.05

In more detail, the experiments were conducted to record and analyze the following points:

- How the number of ports affects the general performance of the system.
- Is there any noticeable difference in the self-assembly architecture.
- In what way the port configuration promotes self-assembly, both regarding the sizes of the different robot groups and the number of groups.
- How the port configuration affects the fitness of the experiment, regarding convergence and the final result.

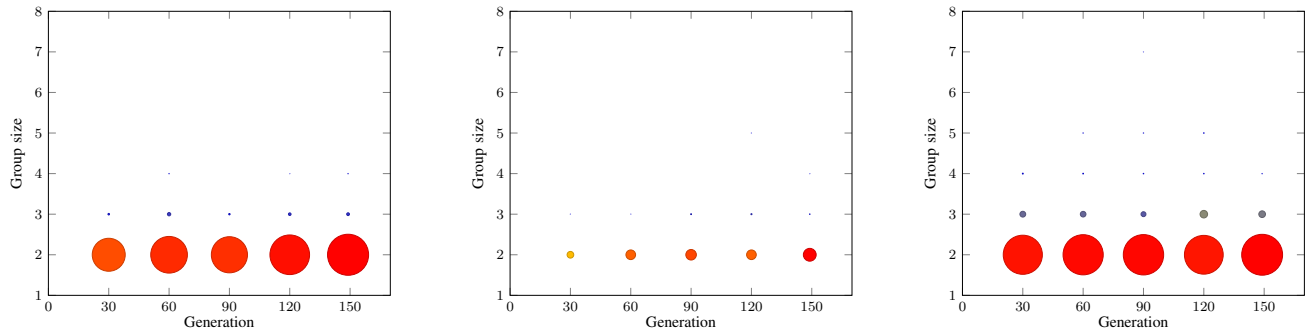
Fig. 4 presents the results for achieved fitness during simulations. The two and four connection port results are very similar ($p - value = 0.15$) where the four connection port performs slightly better with an average fitness of about 0.2 on generation 1 rising to about 0.5 on generation 150. The three connection port simulation is highly different than the two other configurations ($p - value < 0.01$ in both cases). It performs worse, concerning fitness, in every aspect compared to the other port configurations.

The distribution of group sizes formed during simulation is shown in Fig. 5. The size of the circles indicates the number of



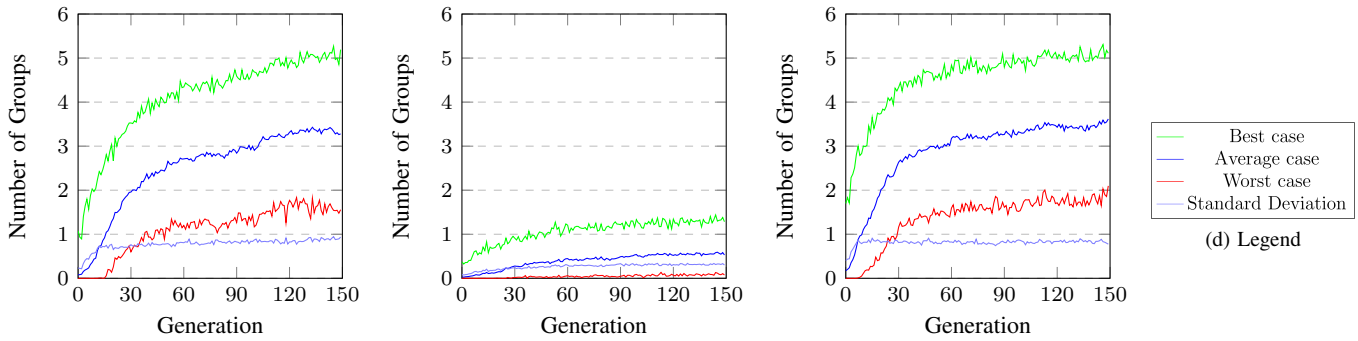
(a) 2-ports (b) 3-ports (c) 4-ports (d) Legend

Fig. 4: Fitness value during simulations.



(a) 2-ports (b) 3-ports (c) 4-ports

Fig. 5: Group distribution during simulations.



(a) 2-ports (b) 3-ports (c) 4-ports (d) Legend

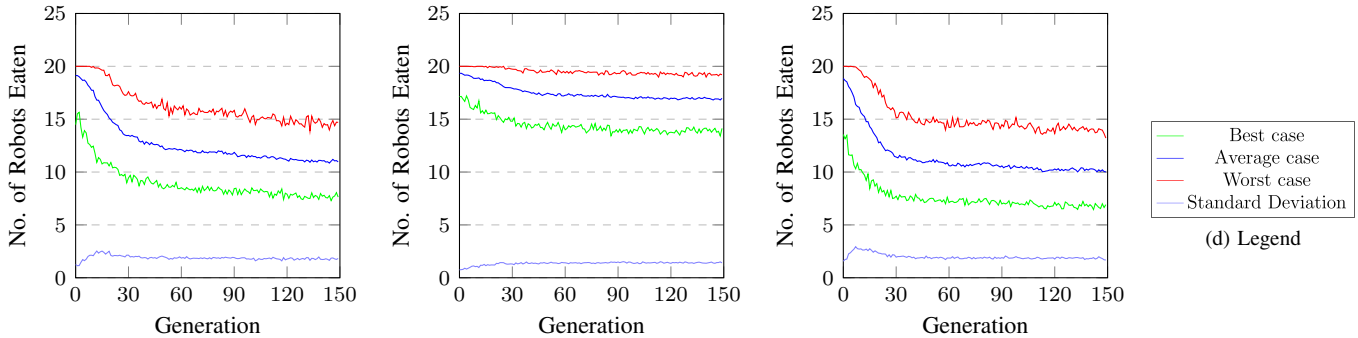
Fig. 6: Number of robot groups formed during simulations.

groups formed. The two and four connection port simulations have more groups of all sizes with the exception of a single group of size five which was formed from the three connection port simulation ($p - value < 0.01$ in both cases). The four connection port simulation formed groups sizes similar to the two connection port simulation ($p - value = 0.36$).

Fig. 6 shows the number of groups which are formed in different simulations. The results for the two and four connection port simulations are very similar ($p - value = 0.21$), where the only notable difference is that the four connection port seems to converge at a faster rate. The three connection port results are quite poor, having few groups throughout the trial. These results are highly different from the two and four connection port results ($p - value < 0.01$ in both cases).

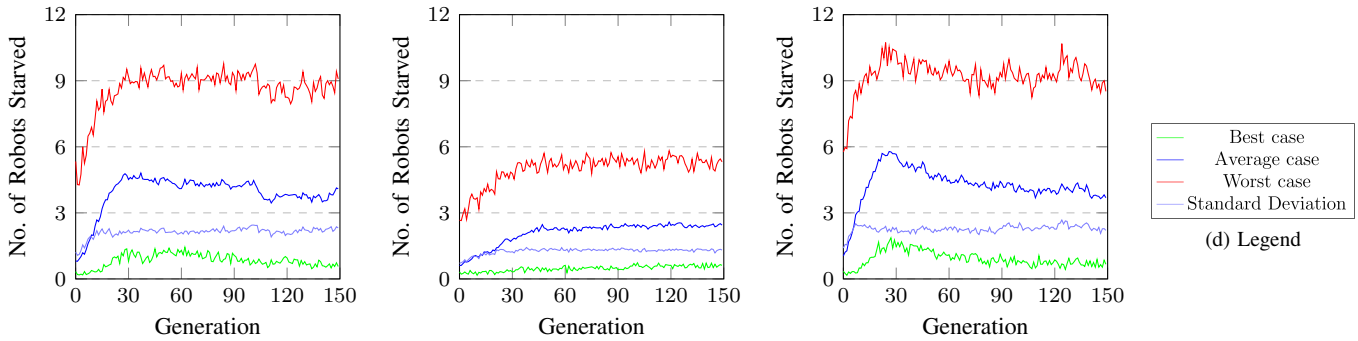
Fig. 7 presents the number of robots which are eaten by predators during simulation. The four connection port simulation performs the best, and is similar to the two connection port robots ($p - value = 0.47$). Where as, the three connection port results are quite poor where a lot more robots are consumed by predators ($p - value < 0.01$ in both cases).

Fig. 8 shows the number of robots starved each generation. The results for two and four connection port simulations are similar ($p - value = 0.10$) where the average number of robots starved is about 4 at the final generation (150).



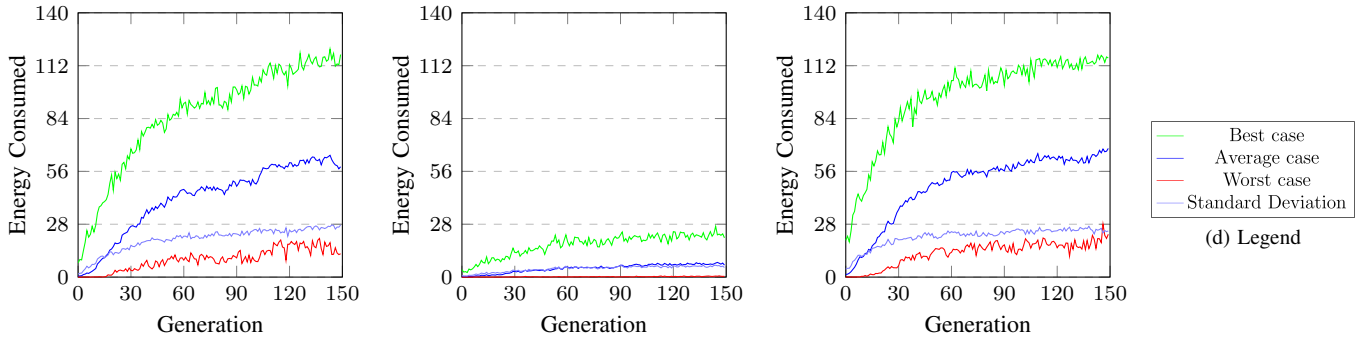
(a) 2-ports (b) 3-ports (c) 4-ports (d) Legend

Fig. 7: Number of robots eaten during simulations.



(a) 2-ports (b) 3-ports (c) 4-ports (d) Legend

Fig. 8: Number of robots starved during simulations.



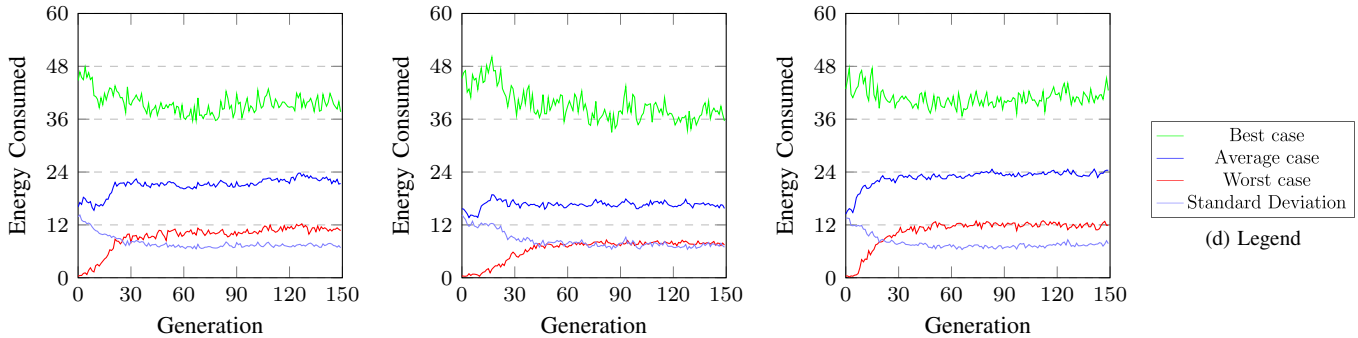
(a) 2-ports (b) 3-ports (c) 4-ports (d) Legend

Fig. 9: Energy consumed by group of robots.

The three connection port simulation is statistically different ($p - value < 0.01$ in both cases) and performs slightly better on these results where the average number of starved robots is slightly less than 3. The three connection port simulation performs the best on this result because most of the robots have been consumed by a predator before they die of starvation.

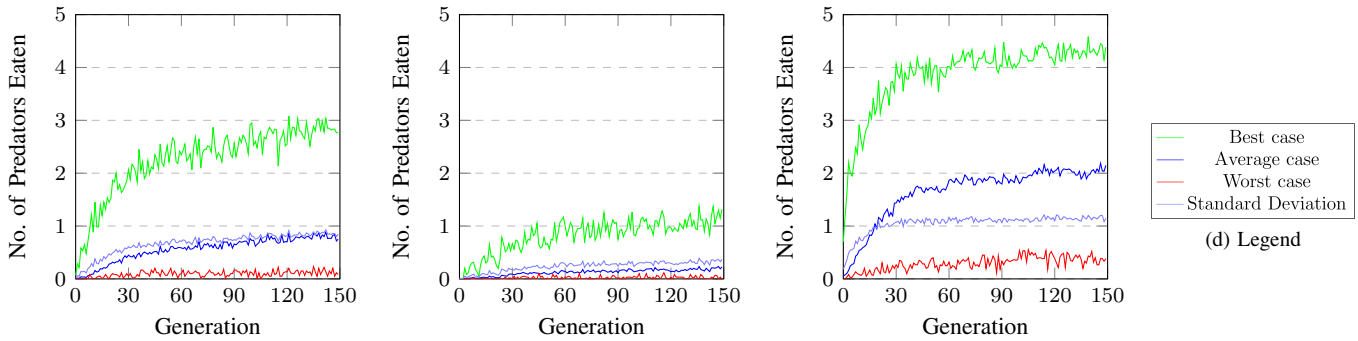
Fig. 9 presents the energy consumed by groups of robots during simulation. Results from two and four connection ports are statistically different ($p - value < 0.01$). Nevertheless, the average for each is of about 60 energy items consumed at generation 150. The three port configuration performs a lot worse with a result of about ten energy items consumed at generation 150 ($p - value < 0.01$ in both cases). The results are correlated with the results obtained from the number of groups formed in the simulation (Fig. 6).

Fig. 10 shows the total amount of energy that is consumed by robots which are not self-assembled. The two and four connection port simulations have similar results and slopes ($p - value = 0.05$). The results for three connection port is statistically different ($p - value < 0.01$ in both cases) and relatively worse. The average values flatten out and do not increase after around generation 30. This is because, more of



(a) 2-ports (b) 3-ports (c) 4-ports (d) Legend

Fig. 10: Energy consumed by robots which are not self-assembled.



(a) 2-ports (b) 3-ports (c) 4-ports (d) Legend

Fig. 11: Number of predators eaten during simulations.

the robots are self-assembling and hence is not tracked as a part of these results. As the energy consumed is not decreasing because more robots are a part of groups, it can be deduced that more energy is consumed on a per robot basis.

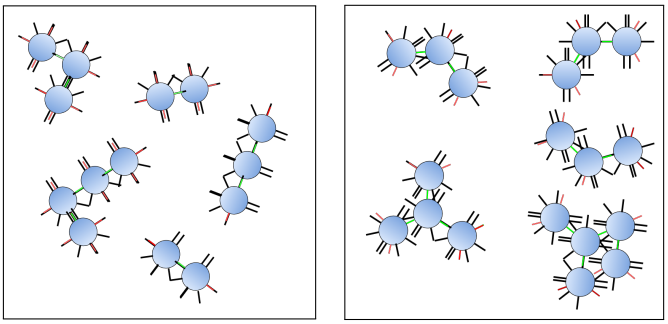
Fig. 11 tracks the number of predators that have been eaten by robot groups. All results are statistically different from each other ($p - value < 0.01$ for each). The four connection port simulation performs the best and is correlated with having larger group sizes than the other port configuration as shown in Fig. 5c. As the robot groups must be of at least size 3 to consume a predator, the results shown in this figure conform with the other results shown earlier.

From the results, it can be observed that the simulations running with a three port configuration are much weaker concerning performance and promotion of self-assembly than the simulations running with two and four connection ports. The reason for this can not be deduced completely from the empirical results. Despite, as the only difference in these simulations are the number of connection ports and the alignment; it is clear that the connection port configuration can significantly impact the performance of the simulation. It can also be deduced that it is not the number of connection ports that has the primary impact on the solution, rather the placement is. This is because the performance using two connection ports and four connection ports are quite similar. If the number of connection ports had a significant impact on the results, we would find the simulation using either two or four connection ports to yield even poorer results than the three connection port simulation. This effect narrows the port configuration problem down to the alignment of the connection ports.

In fact, the robots can rotate their connection ports as a group. A standard strategy which is usually evolved is to either constantly rotate the connection ports in hopes of lining up the ports to another robot, or the robots start rotating their ports when the sensors see another robot in an effort to self-assemble. There are two main problems that the three connection port robots have compared to the other port configurations: (i) none of the port is in the "forward-axis" of the robot, and (ii) the possible group formations by the robots.

It can be seen from Fig. 12 that the different connection port configurations create various types of groups. With two and four connection ports (Fig. 12a), the robot groups take either a line or some square grid formation. Possible formations for the three connection ports robot groups (Fig. 12b) break the pattern of a square grid configuration which makes it hard for other robots trying to connect to the group. The main reason for this connection problem is the relative position a connecting robot needs which is hard to attain because of the large distance among the three connection ports.

There are not significant discrepancies between the results for two and four connection ports. The only result which differs significantly is "predators eaten" (Fig. 11). This is



(a) Two and four connection ports (b) Three connection ports

Fig. 12: Self-assembled robot groups with different assembly combinations.

because, robots with four connection ports tend to form larger groups (Fig. 5). The formation of a greater amount of larger groups naturally agrees with eating more predators, as groups need to be of at least size of 3 to consume a predator. The four connection ports robots attain larger groups because more connection ports allow more points of entry for other robots trying to connect, which increases the probability of succeeding self-assembly to the group. The results also suggest that larger groups do not give rise to better fitness, rather the number of groups (of minimum size 2) correlates with the fitness. This is due to the fact that robots in the port configuration simulations are not in a great need of energy.

V. Conclusion

This work discovers and tests the effects of connection port configuration in a self-assembly system when robots are given basic learning capabilities. The experiment has been conducted using a heavily modified version of the roborobo framework. The results justify that the configuration of the connection ports can significantly impact the emergence of self-assembly using an evolutionary algorithm. The port configuration consists of the number of connection ports each robot has available and the relative positioning of the connection ports on the robot. It is evident from the results that both elements influence the size and frequency of self-assembling robot groups. A possible exploratory point for future work would be to challenge the static nature of the ports considered in this work. Increasing the number of connection ports would be helpful for further analysis. It could be interesting to explore the application of advanced evolutionary algorithms and other bio-inspired algorithms on promoting self-assembly for evolutionary swarm robotics.

References

[1] B. Hlldobler and E. O. Wilson, "The evolution of communal nest-weaving in ants: Steps that may have led to a complicated form of cooperation in weaver ants can be inferred from less advanced behavior in other species," *American Scientist*, vol. 71, no. 5, pp. 490–499, 1983.

[2] G. Vakanas and B. Krafft, "Regulation of the number of spiders participating in collective prey transport in the social spider anelosimus eximius (araneae, theridiidae)," *Comptes rendus biologies*, vol. 327, no. 8, pp. 763–772, 2004.

[3] H. Ulrich and G. Probst, *Self-organization and management of social systems: Insights, promises, doubts, and questions*. Springer Science & Business Media, 2012, vol. 26.

[4] S. A. Kauffman, *The origins of order: Self organization and selection in evolution*. Oxford University Press, USA, 1993.

[5] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *International workshop on swarm robotics*. Springer, 2004, pp. 10–20.

[6] G. M. Whitesides, "Self-Assembly at All Scales," *Science*, vol. 295, no. 5564, pp. 2418–2421, Mar. 2002.

[7] V. Trianni, *Evolutionary Swarm Robotics. Evolving Self-Organising Behaviours in Groups of Autonomous Robots*, ser. Studies in Computational Intelligence. Springer Verlag, Berlin, Germany, 2008, vol. 108.

[8] R. Gross, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous self-assembly in swarm-bots," *Robotics, IEEE Transactions on*, vol. 55, no. 6, pp. 1115–1130, 2006.

[9] V. Trianni, E. Tuci, and M. Dorigo, "Evolving functional self-assembling in a swarm of autonomous robots," *From Animals to Animats*, vol. 8, pp. 405–414, 2004.

[10] P. Varshavskaya, "Distributed reinforcement learning for self-reconfiguring modular robots," Ph.D. dissertation, Massachusetts Institute of Technology, Sep. 2007.

[11] R. Groß, "Self-assembling robots." *KI*, vol. 22, no. 4, pp. 61–63, 2008.

[12] E. Haasdijk, B. Weel, and A. E. Eiben, "Right on the MONEE: combining task-and environment-driven evolution," in *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*. ACM, 2013, pp. 207–214.

[13] N. Bredeche, J.-M. Montanier, W. Liu, and A. F. T. Winfield, "Environment-driven Distributed Evolutionary Adaptation in a Population of Autonomous Robotic Agents," *Mathematical and Computer Modelling of Dynamical Systems, Special Issue: Modelling the swarm analysing biological and engineered swarm systems*, vol. 18, no. 1, pp. 101–129, 2012.

[14] J.-M. Montanier and P. C. Haddow, "Adaptive self-assembly in swarm robotics through environmental bias," in *2014 IEEE International Conference on Evolvable Systems (ICES)*. IEEE, 2014, pp. 187–194.

[15] H. Li, H. Wei, J. Xiao, and T. Wang, "Co-evolution framework of swarm self-assembly robots," *Neurocomputing*, vol. 148, pp. 112–121, Jan. 2015.

[16] N. Bredeche, J.-M. Montanier, B. Weel, and E. Haasdijk, "Roborobo! a fast robot simulator for swarm and collective robotics," *arXiv preprint arXiv:1304.2888*, 2013.

[17] S. Murata and H. Kurokawa, "Self-reconfigurable robots," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, pp. 71–78, 2007.

[18] M. Yim, Y. Zhang, and D. Duff, "Modular robots," *Spectrum, IEEE*, vol. 39, no. 2, pp. 30–34, 2002.

[19] D. Rus, Z. Butler, K. Kotay, and M. Vona, "Self-reconfiguring robots," *Communications of the ACM*, vol. 45, no. 3, pp. 39–45, 2002.

[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 [grand challenges of robotics]," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 1, pp. 43–52, 2007.

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

[22] Y. Zhu, D. Bie, S. Iqbal, X. Wang, Y. Gao, and J. Zhao, "A Simplified Approach to Realize Cellular Automata for UBot Modular Self-Reconfigurable Robots," *Journal of Intelligent & Robotic Systems*, vol. 79, no. 1, pp. 37–54, Jul. 2015.

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

[24] "e-puck Education Robot," http://www.http://www.e-puck.org/, accessed: 2016-06-30.

[25] R. D. Beer, "The dynamics of adaptive behavior: A research program," *Robotics and Autonomous Systems*, vol. 20, no. 2, pp. 257–289, 1997.