Using Hardware Specialization and Hierarchy to Simplify Robotic Swarms

German Espinosa, Michael Rubenstein

***Abstract*— Specialization has always been a tool for work distribution and simplification in nature and in distributed robotics. We present a novel approach to use hardware specialization hierarchically to enhance the capabilities of a swarm without increasing complexity, allowing a numerous group of robots to benefit from the extended features of a few to complete a task that was impossible for them before. We tested the concept under a simulated environment with a classical distributed robotics problem, shape formation, and validated the simulated results against a real experiment.**

I. Introduction

Specialization is a valuable strategy to efficiently solve problems in large groups. Nature provides countless examples of the advantage of differentiating individuals in specific roles: cells[1], bees[2][3], ants[2][3][4], termites[2][3], etc., are often organized in such ways that individuals perform only a small set of tasks as part of a much larger process.

In the past, researchers have used specialization in distributed robotics to increase efficiency or simplify the robot swarm hardware. One common use of specialization is to reduce the group complexity by assigning specific tasks based on the mechanical abilities of different robot types [5]; this "hardware specialization" leverages differences in the hardware of the individuals. This kind of specialization is used in heterogenous swarms, for example allowing reduction of the total number of actuators by avoiding unnecessary redundancies, such in [5], some individuals can drive while others can grab objects. If every individual in the group had the same mechanical configuration, then every individual must have all the required capabilities to complete a goal. This not only means that the final configuration of the robots would be more complex (they would need to be able to drive and grab objects), but also that extending the capabilities of the swarm would require an extension of the capabilities of every single individual.

While counterintuitive, hardware specialization can also be achieved using hardware variability in homogeneous swarms. An example of this could be to let what a priori seems to be a group of identical robots "discover" [6][7] which individuals are better at solving specific tasks based on the success rate attempting to perform them. Then small differences in each robot's hardware performance start playing a part, the individuals accumulate knowledge about their own abilities and use that knowledge to take or abandon specific roles: those with better actuators will move, those with better sensors will sense, for example.

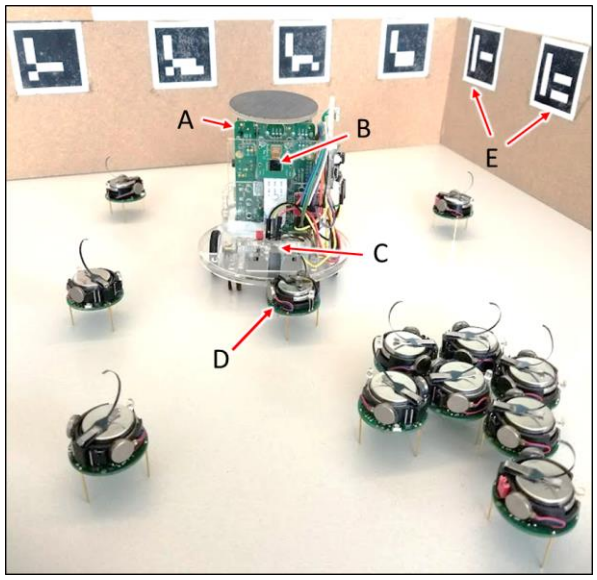

Fig. 1. Coachbot with the raspberry pi board(A), the high resolution camera (B) and the kilobot communication module (C) next to a kilobot (D) and fiducial markers (E).

Another form of specialization, "software specialization", could be achieved by defining roles in the algorithm so, during its execution, the roles will be dynamically assigned to specific agents. An example could be the algorithms that require the selection of a leader to serve as a point of reference for the rest of the robots to achieve a goal or help coordinate the actions of the group. The leader role could be assigned at runtime based on an advantageous condition [8], purely randomly [9] or by hand [10] [11]. A different use of soft specialization is to instruct the robots to "switch" roles when they reach a specific state. A good example are some foraging algorithms where all the participants starts on an explorative mode until the path between the source and the destination is discovered and then some robots switch to a beacon mode and others to a forager mode [12]. Soft specialization can also be found on some sensor network implementations [13], where some sensor-nodes closer to the phenomena under investigation, or with a better chance to get the measurement, specialize in sensing while other nodes take on a sole communication role. The main benefit of "soft specialization" is the possibility to assign roles based of spatial or temporal advantages, such as: when an individual is better located, has more battery, is in range of the goal or became part of a path for example. This also means that if the advantage changes, the roles can be re-shuffled to maintain performance. The main drawback is that every robot must have the hardware abilities to execute all possible roles.

German Espinosa is with the Electrical Engineering and Computer Science Department, McCormick School of Engineering, Northwestern University. (e-mail: germanespinosa@u.northwestern.edu)

Michael Rubenstein is with the Electrical Engineering and Computer Science Department, Mechanical Engineering Department, McCormick School of Engineering, Northwestern University. (e-mail: rubenstein@northwestern.edu).

Even though these uses of specialization allow algorithms to be broken down into roles, most require all participants to have the same level of understanding of the global objective. This means that the complexity of the algorithm is limited by the weakest computing element of the swarm. To overcome this limitation, we propose an alternative use of specialization: the decomposition of a process into "layers" with different complexity, from a complete understanding of the global goal to a very local ability to solve a specific task.

In our layered approach, specific parts of the goal are communicated to the immediate layer below as a mandate to start a task. The recipient of this mandate does not need to know the global goal, and only receives the information required to execute its local task. This interaction between layers is characterized by a hierarchical relationship, where the robots in the lower layers simply execute the commands sent by the upper layers, and the complexity of the goal decreases as the message moves down layers. An important point is that additional information to complete the task can be inferred as an implicit part of the communication between layers. For example, this implicit information could be the time and place the message was communicated. Here, if the top most layer has some sort positional or synchronization information, the rest of the system can "inherit" this knowledge implicitly.

We are looking to show that this approach can help reduce the total complexity of the group by giving the individuals only the computational and sensory abilities to perform in their assigned role and, as an additional benefit, the robots performing the local tasks can have specialized mechanical properties, not required to perform in the other layers.

II. HIERARCHICAL SHAPE FORMATION

A classical problem of distributed robotics was selected to test the value of hierarchy as a form of hardware specialization: shape formation. The technique used was inspired by the growing circles process described in [14], that provides a mechanism to connect global and local rules, but as it is currently not possible to make the robots self-replicate, we decided to combine it with the technique on [15] where robots join the shape after performing stochastic walk when they bump into a growing surface.

The work was divided into two roles: "global" robots that understand the shape and know where circles need to be built and "local" robots that can build and add to the circles.

A high-level idea of the algorithm is as follows: the process begins by providing to the global robots a list of circles, specified by center location and radius, that form the desired shape. Then these global robots will navigate towards where the center of each circle should be located and wait there until it can recruit a randomly walking local robot as a seed. The local robot selected as a seed receives all the parameters needed to form the assigned circle and the global robot moves on to the next circle. The seed stays in place and after a predefined time (called "Initial wait time"), starts recruiting more robots by emitting a decreasing hop-count message that starts at size one and over time increases until it reaches the full size of the circle provided by the global robot. When all circles are fully grown, the shape emerges (fig. 4).

These two layers of work, global and local, demand significantly different abilities. The global role requires positional information, path planning to prevent robot from running into growing circles and global communication to shuffle tasks and share progress information. The local role requires only to perform a random walk and contact-based communication.

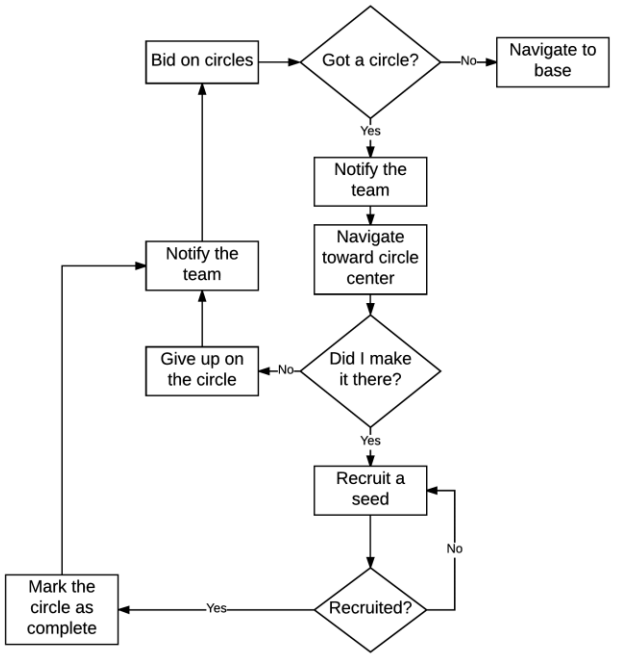

Fig. 2. Program flowchart for global role robots.

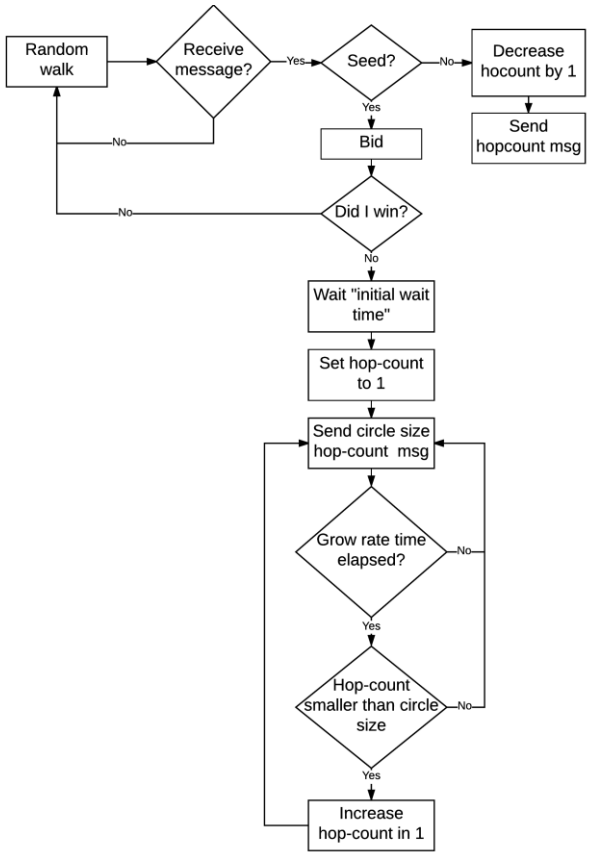

Fig. 3. Program flowchart for local role robots.

One difference between this approach for shape formation and many others using homogeneous swarms [11], is that the global location and orientation of the shape in the environment is completely controlled by the algorithm, this means that no manual seed is required and the resulting shape will be at the exact global position is desired at the end. This is a direct result of the positioning abilities of the global robots and its passed down to the local robots using hierarchy.

III. Roles Definition

The tasks performend by robots in the "global role" can be summarized as follows. Global robots are given a priori a user defined set of circles to be formed, with the location , the size, the priority, the initial wait time and the grow rate of each one. Using communication between global robots, each global robot claims a circle to form based on priority, proximity and availability. Then, using positioning information, they navigate toward the locations where the assigned circle centers should be located and recruit a "local" robot to start building a circle of a specified size and that location (by sending a "seed recruiting" message to the local robot). This interaction can be observed in figure 1. After successful recruitment of a local robot, the global robot communicates to all global robots that the circle was built and bids on a new circle to form (fig. 2). This continues until all the circles have been started.

Each action performed on the circles is transmitted immediately to the rest of the global robots to ensure no circle is assigned twice and all circles are completed at the end of the process.

The "local" robot starts by moving around in the environment using a random walk. If a local robot receives a "seed recruiting" message (from a global robot), the robot will stop moving and bid for the chance to become a seed for a circle. If it fails to become a seed for a circle, due to other robot winning the bid or a faulty communication, it will return to moving randomly. In case of winning the bid, the seed will wait the time specified as the "initial wait time" and then start transmitting a "growing circle" message with a hop-count one. From that moment on, it will increase the hop-count by one whenever the time specified as the "growing rate" elapses until it reaches the size of the circle. All these parameters: the circle size, the initial wait time and the grow rate, are given to the seed robot by the global robot during the recruitment.

If a local robot walking randomly receives a "growing circle" message with a hop-count value greater than zero, it will stop moving, reduce the hop-count value by one and retransmit this message (fig 3).

IV. Selected hardware

The Kilobot platform allows the use of a large number of robots that are easy to operate collectively and also meets all the requirements of the local role, as it can perform a random walk, and has short range communication and distance sensing. These last two can be used together to simulate a downgrade to contact-based communications like it is needed for the algorithm. For more information about kilobots, see [9].

The global role needed a robot platform with the ability to sense its position, plan paths and move without colliding with other global robots. In terms of communications, it required the ability to send progress information to its peers at all time and also short-range communication with nearby kilobots. A custom designed robot called "Coachbot" was built to meet these needs. Coachbot is a two-wheel differential drive robot built within a circular acrylic frame. The frame is designed to run in an environment shared with kilobots by being able to gently push through a crowd of them without knocking them over. Each Coachbot is powered by a 3.7V 2Ah rechargeable battery and an onboard raspberry Pi 3 rev 1.2 computer with a 1Ghz ARM processor and built in WIFI, a high-resolution forward-facing camera and a special communication module attached in the front to send and receive messages to and from touching kilobots. To share progress and positioning information between Coachbots, they send peer to peer messages over WIFI. This distributed communication system allows changes in the number of Coachbots during the execution of the experiment without impacting the result. The coachbot total part cost is under $120.

To get positioning information, Coachbots use a number of Aruco [16] fiducial tags located in the walls of the arena. Each tag is uniquely identified and the global position of each tag is known by all Coachbots. Using its camera, Coachbot can compute the relative position of the robot respect to the tags captured by the camera, and therefore estimate the location and orientation of the robot. Aggregating information from multiple tags reduce the impact of noise in the readings and improve the accuracy in the positioning information produced. For this reason, we incorporated multiple tags in different sizes in every possible view angle (fig. 1).

V. Simulation environment

To test the performance of the shape formation, we

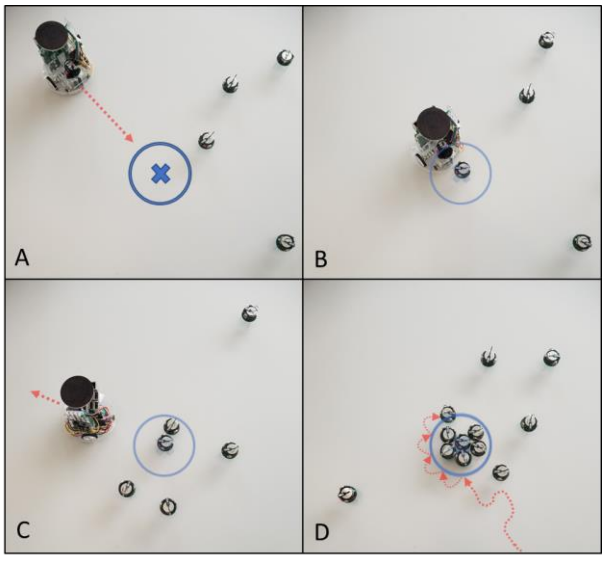

Fig 4: (A) Coachbot moving toward the center of the assigned circle of hop-count one. (B) Coachbot recruiting a kilobot to serve as the seed for the circle. (C) Coachbot moving away toward the next circle and the seed starting the circle formation. (D) Circle formation in advanced state, randomly walking kilobot performing edge following.

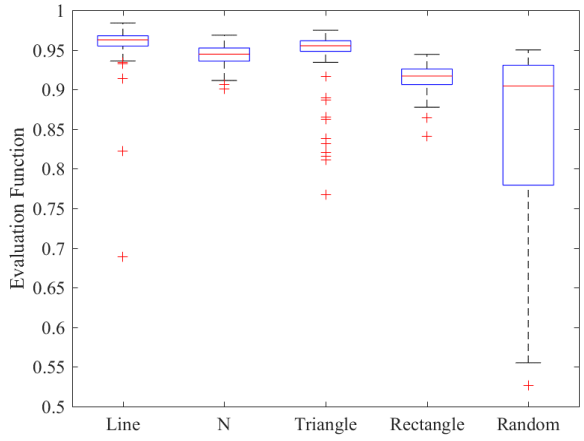

Fig. 5. Value of evaluation function, by shape. Computed from 100 simulations per scenario.

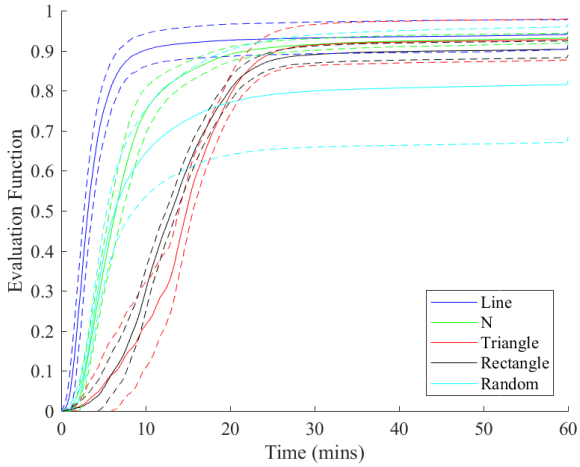

Fig. 6. Value of evaluation function over time, by shape. Computed from 100 simulations per scenario. Solid line represents the mean and dotted line the standard deviation.

developed a simulated environment with all the features required in the shape formation process. A virtual three meters by three meters arena hosted one thousand robots for the local role and up to four robots for the global role. Robots were represented by a circular shape that cannot overlap and cannot move outside of the arena. All robots started in a randomized position and ran for sixty simulated minutes. The program simulates the movement and communication behaviors needed for the local and global roles robots. Collision detection was only added to the local robots, as in the real experiment the Coachbots hardware was designed to be able to push its way through kilobots and they will use positioning and planning to avoid each other. All the results showed in section VII and VIII were obtained by repeating simulated experiment 100 times for each parameter variation.

VI. Evaluation Function

To quantitatively measure how well the shape is formed with the described hierarchical shape formation algorithm, a custom evaluation function was used. The evaluation function is given the desired shape and the position of all robots at a particular time. It then reports the ratio of area inside the shape that is farther than one Kilobot diameter from any other robot, these are locations within the shape that another Kilobot could have been added. Figure 14 shows in red an example of the area considered by the evaluation function farther than one Kilobot diameter a way from other robots in the shape.

Each robot counts not only by its own surface but also has a small influence area around it where there is not enough space to fit another robot. The areas within the desired shape where there is enough space to fit a robot but it is not occupied by any, it is discounted from the area of the shape. The evaluation function then reports the ratio of space where no other robot will fit to the total area in the shape.

VII. Shapes and Simulation Results

It was necessary to show the algorithm worked well with various desired shapes, such as contours and solid ones, to do so, two experiments of each kind were selected: a straight line and an "N" will test contours, and a filled triangle and a rectangle will do the same for solid shapes. In addition, to ensure the method is nonspecific to these four, the algorithm was tested with 100 random shapes (fig. 7). Each random shape was formed by a group of 18 circles of sizes 1, 2 and 3, randomly distributed in a 1m by 1m section of the arena to maximize overlapping. In terms of shape completion, the results show that the algorithm performance is better in contours than solid shapes and is affected by the overlapping area of circles and the sharpness of contours (fig. 5).

In terms of speed of formation, the solid shapes took longer to build. The straight line was the fastest, and did not improve much after the first ten minutes. The N contour had a significant change in the slope at ten minutes but continued to improve through the following twenty minutes of the experiment. In both solid shapes, it can be appreciated the wait times to trigger the circle formation as changes in the curve slope. Random shapes where the slowest, taking close to thirty minutes to reach an average value of 0.8 (fig. 6).

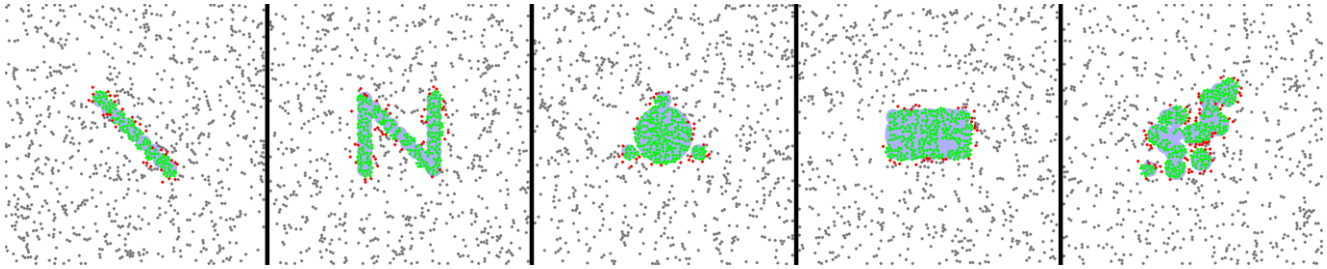

Fig. 7. Final robot configuration and desired shapes. Evaluation function results from left to right: Line 96.89%, "N" 93.77%, Triangle 95.56%, Rectangle 89.58% and Random 92.10%. In green shows robot that are part of the shape, in red robots performing edge following and in gray robots that are in random walk state. The desired shape is marked with a light blue shade.

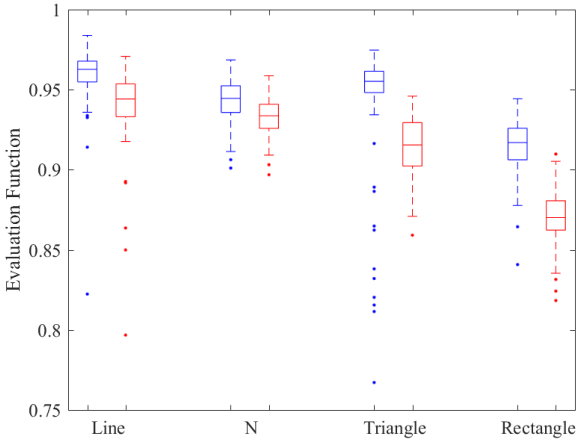

Fig. 8. Value of evaluation function, with (blue) and without (red) edge following, by shape. Computed from 100 simulations per scenario.

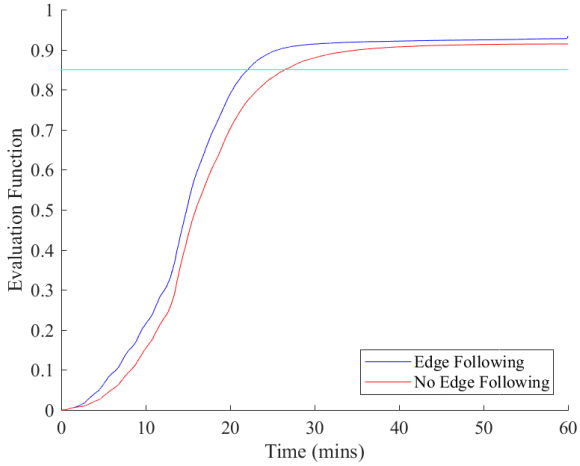

Fig. 9. Value of evaluation function over time, with and without edge following. Computed from 100 simulations per scenario.

VIII. Implementation Adjustments

By analyzing the initial results, some additional parameters and behaviors that increase the shape completion and speed up the process were identified.

Edge following: The local robots originally only joined the shape when they randomly ran into another robot sending a "growing circle" message with a hop-count bigger than zero, otherwise they just kept walking randomly. By adding edge following, the kilobots follow the edge of the shape (where the hop-count is zero) once they bumped into it, instead of staying in the random walk behavior. This allows them to try to find a spot where they can join a circle, even if they arrive to the shape in a non-growing area (fig. 4). The change in the local role increases the chance of a kilobot of joining a growing shape and produce better and faster results. To evaluate the improvement, we performed the same shape formation simulations with and without edge following behavior and compared the output. The results showed that edge following improvement is more significant in solid filled shapes. The median formation in triangles improved around 4% and in rectangles improved over 5% (fig. 8).

Edge following also increased the speed of the algorithm by keeping robots near the shape while the circles are forming and, when the hop-count of the circle is incremented during the formation, all robots performing edge following are likely to be recruited by the circle immediately. Comparing the completion over time, on average the formation was five minutes quicker to reach value 0.85 for the rectangle shape once edge following was implemented (fig. 9).

Initial wait time: One complication that can arise with this algorithm is that a circle can be blocked by others surrounding it before it is complete. This happened mainly when complex shapes have circles located close enough to each other to block the way of nearby walking kilobots. To prevent this problem, we defined a parameter per circle to specify a wait time for the seeds before starting sending hop-count messages. This allow circles surrounded by other circles to start growing before the circles around it can block it, generating a more consistent and better result. By waiting 5 minutes before every circle started the recruitment in a 5x5 lattice of circles, the evaluation function showed a significant change allowing the top 75% of the executions to improve from at least value 0.74 to 0.78 (fig. 10).

Grow rate: Another initial complication occurred in experiments with bigger circles, where growing surfaces with a high hop-count tended to have diffusion limited aggregation problems [17] and form long branches with lots of empty space. These structures blocked the way for walking kilobots and prevented these circles from achieving a high density. To

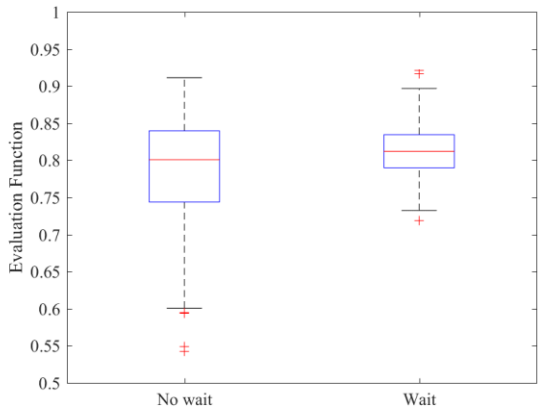

Fig. 10. Value of evaluation function, with and without initial wait time. Computed from 100 simulations per scenario.

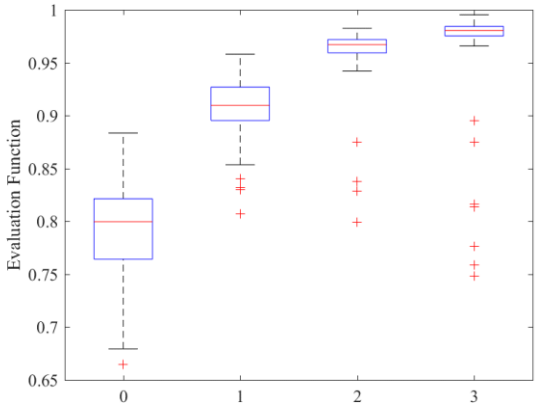

Fig. 11. Value of evaluation function, by grow rate in minutes per hop-count increase. Computed from 100 simulations per scenario.

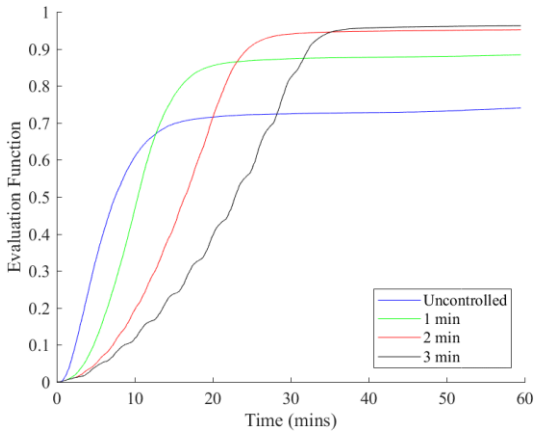

Fig. 12. Value of evaluation function over time, by grow rate in minutes per hop-count increase. Computed from 100 simulations per scenario.

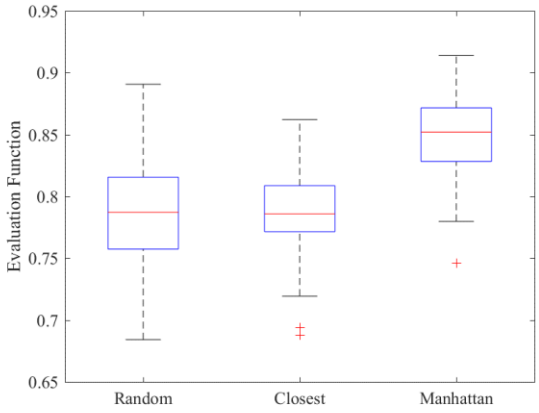

Fig. 13. Value of evaluation function, by circle priority strategy. Computed from 100 simulations per scenario.

avoid this problem, we added another parameter to the circles to control the growth rate. The seed, instead of starting the hop-count with the circle size, starts at zero and increases the hop-count at a fixed rate until reaching the desired value at a configurable pace. By controlling the time between hop-count increases, the completion improved on large circles, reducing the median incomplete area from 0.2 to 0.02 in circles of hop-count 10 (fig 11). Figure 12 shows the correlation between completion and the grow rate. Each one of the three increments in the grow rate, improved the results.

Priority: Originally the coachbots assigned the circles only based on the global robot's current distance to the circle. This solution works fine for simple contours, but proved to be problematic with complex shapes, especially on solid ones. The main drawback is that in a solid shape, is necessary to start building it on a specific order to prevent isolating pending circles. An example of this can be a rectangular lattice like shape, if the circles on the edge are recruited first, then accessing the inner ones becomes difficult or impossible. To address this problem, we added a priority parameter to the circles based on the Manhattan distance of the circle to the center of the shape, giving the interior circles higher priority. We found that by doing that we could improve the median results by 6% in solid complex shapes (fig. 13).

IX. Experimental Setup

The results of a smaller experiment with real robots was compared against the same configuration in the simulation to validate the accuracy of the data obtained from the simulation. Each experiment ran for a maximum of fifteen minutes forming five circles (4 with size 1 and 1 with size 3) in a diamond formation. The experiments were considered finished when there was no growing surface exposed to randomly walking local robots or when the time was up.

The experiment was performed ten times in a one-meter by one-meter arena with 100 kilobots and 2 coachbots. Kilobots cannot sense the edge of the arena, so ones touching the edge were rotated by hand to allow them to continue to move randomly. To capture the activity during the experiments, a ceiling camera that took one picture per second was installed and a computer vision algorithm computed the location and the state of each robot in the arena. After the data was captured, it is then used to compute the completion rate using the evaluation function.

X. Experimental Results

The average completion value of the experiment was 0.95, very close to the 0.94 obtained by the simulation with the same setup. Surprisingly, both environments reached 0.85 average value at a very similar time, around 6 minutes and 45 seconds.

One observation was that, like in a pixelated image, in a reduced number of circles, small variations in the location of the circles had a great effect in the recognizability of the desired shape in both, the simulation and the real experiment. However, the circle formation performed as predicted in the simulation and the metrics from the evaluation function have shown a high completion value.

Another observation from performing the experiment in real robots was the realization on how the physical interaction was influencing the results in a way that was not fully modeled in the simulation. Randomly walking and edge following robots bumped into non-growing areas of circles and pushed other stationary robots altering the formation. This unaccounted interaction caused two different effects: one detrimental and one beneficial to the shape formation. When the seeds of the circles are by themselves (during the initial wait time) or the circles are in a very early state and do not have enough mass to withstand the collisions, the bumping caused migrations of the circles centers. This caused robots to end the experiment outside of the desired shape and do not count for the evaluation function. In the other hand, when the circles are more complete, the edge following behavior of the robots running into the shape, creates an inward movement in the edge of the shape reducing the empty space and increasing the number of robots forming each circle, making circles more compact (fig 14). This unaccounted phenomenon caused the completion rate to increase and decrease over time, when in the simulation it only increased (fig 15).

Figure 15 also shows the correlation in the results from the both, the physical experiment and simulated one. This information supports the hypothesis than the experiment will behave close to the simulated results when performed in bigger scenarios.

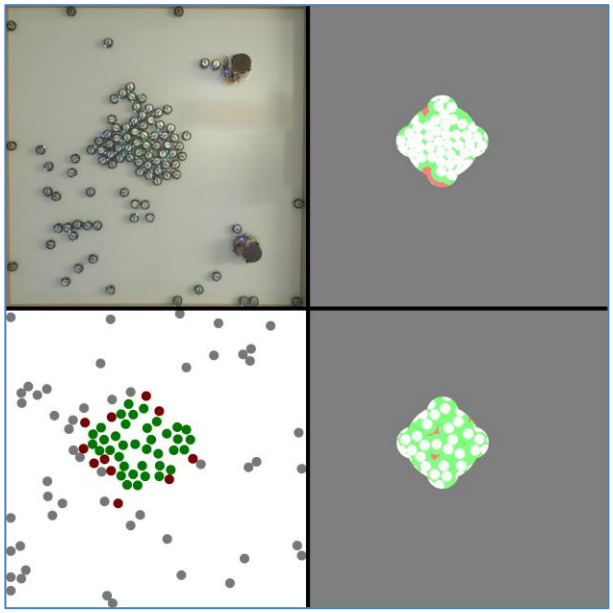

Fig. 14. Screen shot from experiment and simulation execution with associated evaluation function.

XI. Conclusion and Future Work

In this paper, we showed a novel approach for specialization using hierarchy to reduce the complexity of robot swarms. The fundamental idea we proposed, is to divide work into layers, from a global understanding of the goal to a local knowledge of a task. We showed supporting evidence that the approach allowed a simple swarm to achieve an objective it was not able to achieve before by adding a global role assigned to a more complex robot to help support the process. We performed simulated experiments to validate the process and improve the settings of the different parameters in the algorithm.

The described use of specialization and hierarchy in the swarm allowed the Kilobots to operate without distance sensing, using contact based communication. This reduction in needed capability could result in approximately $2.50 cost savings per Kilobot (the cost of distance sensing hardware), and therefore $2500 for the whole 1000 robot swarm. This savings is an order of magnitude more than the cost of adding a few Coachbots used to reduce the needed Kilobot capabilities. Using cost as an approximation of swarm complexity, this means the use of specialization and hierarchy allows for a significant reduction in swarm complexity when compared to similar tasks in a homogenous swarm, such as in [11]. In addition, the shape formation task presented here is greatly speed up using the coachbots, forming shapes in approximately 30 min compared to 12 hours in [11] and also requires no human intervention to secure the location and orientation of the shape in the arena.

We tested the concept using it only for shape formation, however, we believe this separation of work and duties could be applied to other problems. Future work includes the test of hierarchy in other scenarios like collective transport and foraging, as well as to approach complicated problems like algorithm healing and fault tolerance.

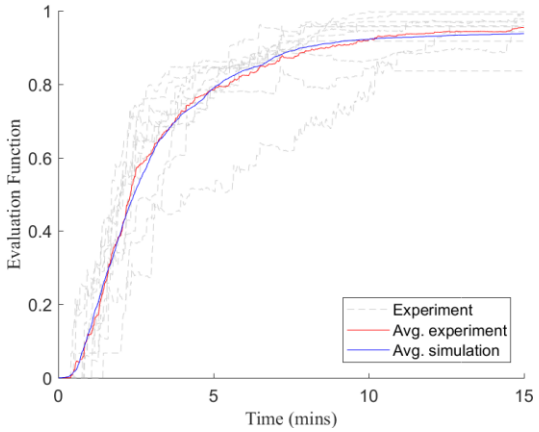

Fig. 15. Evaluation function over time in minutes: Simulation Vs Experiment. Computed from 100 simulations and 10 experiments.

References

[1] Ispolatov et al. "Division of labour and the evolution of multicellularity." Proc. of the Royal Society of London B: Biological Sciences 279, no. 1734 (2012): 1768-1776.

[2] Robinson, Gene E. "Regulation of division of labor in insect societies." Annual review of entomology 37, no. 1 (1992): 637-665.

[3] Beshers, S. N., and J. H. Fewell. "Models of division of labor in social insects." Annual review of entomology 46, no. 1 (2001): 413-440.

[4] Wilson, Edward O. "Division of labor in fire ants based on physical castes (Hymenoptera: Formicidae: Solenopsis)." Journal of the Kansas Entomological Society (1978): 615-636.

[5] Dorigo et al. "Swarmanoid: a novel concept for the study of heterogeneous robotic swarms." IEEE Robotics & Automation Magazine 20, no. 4 (2013): 60-71.

[6] Labella et al. "Division of labor in a group of robots inspired by ants' foraging behavior." ACM Transactions on Autonomous and Adaptive Systems (TAAS) 1, no. 1 (2006): 4-25.

[7] Labella et al. "Self-organised task allocation in a group of robots." In Distributed Autonomous Robotic Systems 6, 2007.

[8] Stilwell et al. "Toward the development of a material transport system using swarms of ant-like robots." In IEEE International Conference on Robotics and Automation, 1993.

[9] Rubenstein et al. "Kilobot: A low cost scalable robot system for collective behaviors." In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 3293-3298. IEEE, 2012.

[10] Chen et al "Segregation in swarms of e-puck robots based on the brazil nut effect" International Conference on Autonomous Agents and Multiagent Systems, 2012.

[11] Rubenstein et al. "Programmable self-assembly in a thousand-robot swarm." Science 345, no. 6198 (2014): 795-799.

[12] Hrolenok et al. "Collaborative foraging using beacons." International Conference on Autonomous Agents and Multiagent Systems, 2010.

[13] Byers, John, and Gabriel Nasser. "Utility-based decision-making in wireless sensor networks." First Annual Workshop on Mobile and Ad Hoc Networking and Computing, 2000.

[14] Kondacs, Attila. "Biologically-inspired self-assembly of two-dimensional shapes using global-to-local compilation." In 18th international joint conference on Artificial intelligence, 2003.

[15] Neubert et al. "A robotic module for stochastic fluidic assembly of 3D self-reconfiguring structures." In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 2479-2484. IEEE, 2010.

[16] Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion." Pattern Recognition 47, no. 6 (2014): 2280-2292.

[17] Witten, Thomas A., and Leonard M. Sander. "Diffusion-limited aggregation." Physical Review B 27, no. 9 (1983): 5686.