


UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project
Final Paper 2022

Title: Evolving Problem-Solving Collective Behaviour and Functional Diversity in Swarm Robotics

Author: Bailey Green

Project Abbreviation: Swarm

Supervisor(s): Geoff Nitschke

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	5
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	0
Total marks		**80**	**80**

ABSTRACT

Swarm robotics is a promising field which stems from the field of swarm intelligence. Swarm robotics has lots of potential and ongoing research to try to find the most effective methods for solving tasks. This paper looks at comparing two such methods that makes use of evolutionary algorithms to solve a simple collective construction task in a virtual environment. The first method is mEDEA which is an environment-driven evolutionary algorithm that can react well to unknown or changing environments but somewhat lacks in providing diverse as well as high performing solutions. The second method is a hybridized method using mEDEA as a baseline. The mEDEA method was combined with novelty search and local competition which is a behaviour-based method to try to improve the diversity and performance of the base mEDEA method. No significant difference was found between the mEDEA method and our hybrid method for this particular task. Both methods performed to an equal high standard, and we believe our hybrid method could still provide benefits in more complex tasks.

1 INTRODUCTION

Swarm intelligence follows the behaviours of a group of interacting agents. Each individual in the group may be considered unintelligent but the system as a whole can behave in some form of collective intelligence. Swarm intelligence can be seen in both nature as well as artificial systems. It is from nature that many of the algorithms for artificial systems have been inspired. Many of these swarm intelligence algorithms have similar properties such as information sharing among multiple agents, self-organisation and co-evolution among agents, high efficiency for co-learning, and they can be easily parallelised for practical and real-time problems [8]. Two examples of common nature inspired swarm intelligence methods are particle swarm optimisation (PSO) and ant colony optimisation (ACO) [13].

Evolutionary Algorithms is another technique inspired from natural evolution and Darwin's survival of the fittest strategy [4]. It is the baseline principles for some of the methods and techniques that will be described in section 3 of this paper. We go through how a typical evolutionary algorithm works in section 2.1. Simple evolutionary algorithms are often limited by slow convergence properties and limited diversity. More advanced methods are required to overcome this limitation.

Swarm robotics is the main focus of this paper. Swarm robotics is an application of swarm intelligence. It is a research field in which a group of robots work together in order to perform some task [7]. The robots do this by both interacting with the environment as well as other robots within a certain proximity to them through simple sensors and communication devices, similar to the way in which a group of animals may act. These robots can either be homogenous or heterogenous. They are typically homogenous agents with low complexity. These robots also typically operate in a decentralized manner and so no central control system is needed. The way in which these robots are controlled is rather simple. Data is read in from the various sensors of the robot. This data is then processed in some way and used to control the robot's output which will be connected to motors that may control the robot's wheels. These robots can either be physical robots interacting with real world environments or can be simulated robots interacting with a setup virtual environment. Most physical swarm robotic systems do not have adaptive behaviour methods whereas the simulated systems are mostly used as an experimental platform for adaptive behaviour methods. In this paper we will be using simulated robots acting in a virtual environment.

Common swarm robotics tasks solved using an evolutionary approach include foraging [18], aggregation [19], hole avoidance [20], categorization [15] and group transport [17]. All of these applications require an adaptable solution that cannot be easily achieved using more traditional approaches. These tasks can also often be dangerous or less effective to use humans and so swarm robotics is a better solution.

Solving problems using swarm robotics is a complex task and is something that has lots of ongoing research. The main three areas that are looked at when using a method to solve a swarm robotics problem are: being adaptable to unknown or changing environments, being able to achieve a high maximum fitness value to be able to solve the task effectively and having a diverse swarm.

1.1 Research Objective

The primary research objective for the paper is determining the best method for evolving problem-solving collective behaviour and functional diversity in swarm robotics. Two methods that make use of evolutionary algorithms will be tested and compared in a collective gathering task. The first method will use the mEDEA algorithm. The mEDEA algorithm is an environment-driven evolutionary distributed evolutionary adaptation algorithm and should provide the ability to adapt to unknown or changing environments as needed. While mEDEA provides a good method for unknown or changing environments and can provide a relatively diverse set of solutions it is shown to have relatively low fitness values [2]. We would also ideally want to maximise the fitness and diversity while still providing environment adaptation. This is the motivation behind our hybridized method. We go more in depth on the mEDEA algorithm is section 2.3.

The second method will be the hybridized method of the mEDEA algorithm with novelty search and local competition. Novelty search is a behaviour based evolutionary algorithm that is driven by the novelty of a behaviour [10]. Novelty search is used in combination with local competition to reward agents

Algorithm 1 Evolutionary Algorithm [21]

Initialization: Generate an initial population

While: Stopping conditions are not satisfied **do**

Evaluate all individuals in the population

Select all individuals in the population

Apply generic operators(crossover, mutation to

Generate offsprings

Replace a proportion or the entire population with

Offsprings

End While

Algorithm 1: Showing pseudo code for a typical evolutionary algorithm [21]

outperforming other agents of a similar type. The reason for using novelty search with local competition is that it has shown to be able to achieve a high diversity of solutions to a problem while still maintaining high fitness values [11]. While novelty search has been tested with swarm robotic systems [5], we could not find any research of testing novelty search and local competition in a swarm robotics system. Novelty search and local competition is described in more detail in section 2.4 and 2.5. We hypothesize that the hybridized method will have the environment adaptability of the mEDEA method as well as the ability to achieve a high maximum fitness value and a diverse swarm. The efficacy of the hybrid method will be tested against mEDEA in a range of increasingly complex collective behaviour tasks.

2 BACKGROUND & RELATED WORK

The following section contains the necessary background information for understanding the two methods we will be comparing in this paper. We also look at other related work to show what else has been done and provide motivation for using our particular hybrid method.

2.1 Evolutionary Algorithms

Evolutionary algorithms work by using the foundations of evolution and DNA. As shown by the pseudo code in algorithm 1, initially it begins with a randomly generated population which is a set of solutions to the problem trying to be solved. Each solution is represented by a genotype. The performance of the solution is then represented by a fitness value. The higher this fitness value is the more chance the solution has of surviving into the next generation. At the end of each generation a certain number of solutions are chosen to form the population for the next generation based off their fitness values. Once this selection has taken place a few of these solutions may be selected for reproduction to generate new solutions. This child solution that is generated from two parent solutions may then have a random mutation applied to it before being placed in the population for the next generation. These children will replace either all or part of the previous generation depending on the algorithm. This process is then repeated until some stopping condition is satisfied.

2.2 Evolutionary Robotics

Evolutionary robotics is a field that uses evolutionary computing and evolutionary algorithms to solve tasks using robotics [16]. It is a technique that can be used for the design and control of swarm robotic systems using the selection, mutation, and replacement principles of natural evolution. It is powerful in that you do not need to manually compose a solution to a specific problem and provides robust and adaptable solutions to many tasks. Due to the use of evolutionary computing solutions are iteratively generated.

Evolutionary robotics has been widely researched and there a few key things that are commonly accepted in the field [16]. The first is that neural networks make a good controller paradigm for the robots. Feed forward neural networks are often used, receiving inputs from the robot's sensors and then having outputs to control the robot. It is the evolutionary approach that will modify the weight values of these neural networks to control the robots. Another thing we know is that relying on objective-based fitness can be misleading. It often does not lead to very diverse solutions and can get stuck in a local maximum. It is this reason that we have chosen the behaviour-based approach of novelty search with local competition to be used in our hybridized method. The final common thing we know is that applying these techniques to real robots can be challenging and does not always map perfectly from the simulated to the real world. This paper will only focus on the simulated world and so will not have to deal with the complexities of applying these techniques to the real world. The goal is only to see if the hybridized method would be a viable solution to a swarm robotic problem and if it can improve upon the mEDEA method.

2.3 mEDEA Algorithm

There can be many cases in swarm robotics, and robotics in general where you are faced with an unknown or constantly changing environment and need a way to deal with this. Environment-driven distributed evolutionary adaptation (EDEA) is one way of doing this. EDEA relies on the fitness function which is the result of two motivations. The one motivation is the extrinsic motivation where agents must optimise the interaction between itself and the environment and possibly other agents in order to maximise its survival. The other intrinsic motivation is the sharing of genomes through the close interaction of agents. The larger the number of agents a robot is able to meet, the greater the chance of survival. These two motivations need to be balanced correctly to ensure an efficient EDEA algorithm and maximum genome sharing as well as environment interactions [2].

The mEDEA algorithm (a minimal EDEA algorithm) is an algorithm that takes the factors mentioned above into account. As shown by the pseudo code in algorithm 2, this algorithm is

Algorithm 2 The mEDEA algorithm [2]

```
genome.randomInitialize()
while: forever do
    if genome.notEmpty() then
        agent.load(genome)
    end if
    for iteration = 0 to lifetime do
        if agent.energy > 0 and genome.notEmpty() then
            agent.move()
            broadcast(genome)
        end if
    end for
    genome.empty()
    if genomeList.size > 0 then
        genome = applyVariation(select_Random(genomeList))
    end if
    genomeList.empty()
end while
```

Algorithm 2: mEDEA algorithm [2]

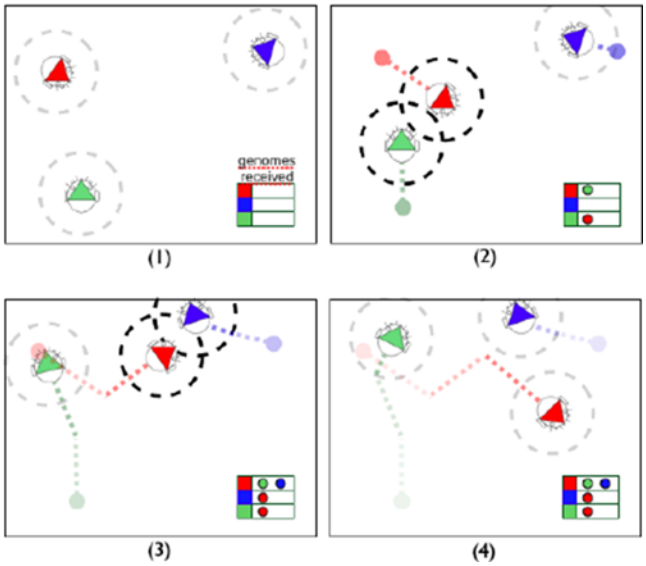

Figure 1: Showing agents broadcasting and receiving genome information from other agents in range [2]

implemented by having a genome which is controlling an agent throughout a generation. Initially this genome is randomly initialized. Each generation the genome is loaded onto the agent and as long as that genome is not empty and the agent has energy, the agent will move around the environment and broadcast its genome to other agents in a certain proximity. This is shown in figure 1 during step 2 where the red and green agents come into close proximity with each other and share their genomes and again in step 3 with the red and blue agents. At the end of each generation the current genome is emptied. There are then three operators that are used to find a new active genome for each agent. The selection operator finds a random subset from the list of genomes the agent has received from other agents. The variation operator then applies some mutation to these genomes in order to modify them slightly. Usually, this modification is very minor. Finally, a replacement operator is used to randomly select the new active genome to control the agent for the next generation and the list of genomes the agent received is emptied [2].

The mEDEA algorithm works well for reacting to unknown or changing environments but not as much for task-based evolution. Our hybridized method will make use of the mEDEA algorithm as a base for the environment adaptation but will be combined with novelty search and local competition to hopefully provide more diverse and better performing solutions. There have been other studies that have hybridized the mEDEA algorithm with other behaviour-based methods.

A study was done to try integrate the environment-driven evolution approach of mEDEA with task-driven selection. The task given to the robots was to collect pucks spread around an arena. The more pucks they collected, the more credits they earned. The results showed that using their algorithm the robots collected far more pucks over the mEDEA algorithm. Their algorithm also did not have any substantial effect on the environmental adaptability of the mEDEA algorithm [6].

Another study done comparing the mEDEA algorithm to various embodied distributed quality diversity algorithms (EDQD) which are a variation of the mEDEA algorithm [9]. An experiment was done where again the robots had to go around a circular arena and collect red and blue tokens. Once a token is consumed a new token of the same colour is generated at a new location. The fitness function was defined as the total number of tokens collected during a lifetime. The results show the EDQD algorithms were able to collect more tokens than the mEDEA algorithm. The EDQD algorithms also seemed to explore and react to environment changes better than the mEDEA algorithm. This is due to the higher level of functional diversity that was explored using the variations of the EDQD algorithms.

2.4 Novelty Search

Typically, in an evolutionary algorithm the fitness function is driven by some objective. Novelty search differs from this in that the fitness function is goal independent and is rather driven by the novelty of a behavior. A study was done looking at the limitations of objective-based search and comparing it to novelty search. The results showed that in both a navigation task as well as a more difficult biped robot task, novelty search was able to significantly

outperform an objective-based search [10]. There are two main motivations for using novelty search [3]. The first being obtaining a particular goal space as quickly and frequently as possible [10]. The second motivation for using novelty search relates to quality diversity, where we want to generate solutions that are as diverse as possible [12].

Novelty search can be implemented rather easily by simply replacing the objective-based fitness function in the evolutionary algorithm with some measure that quantifies how different the behavior of an individual is with respect to other previously identified individuals' behaviors. These previous behaviors are stored in some data structure (called the archive) that is initially empty. If a behavior is significantly different from the ones already in the archive (above some threshold) that behavior is added to the archive [5].

A study was done in which two experiments were conducted in order to see how well novelty search performed in two swarm robotic tasks compared to an objective-based approach [5]. The first experiment was a swarm aggregation task in which a scattered robot swarm must form a single cluster. Both the objective-based search and novelty search achieved a similar level of fitness scores. It was also found that novelty search was able to find several alternative and successful solutions to this task. The second experiment was a resource sharing task in which the swarm must share a single battery charging station. The resource should be shared efficiently to ensure each robot gets sufficient time to maximise survival. The results of this experiment showed that the objective-based approach often got stuck in a local maxima. As a result of this, novelty search was able to perform significantly better in this task. The solutions it was able to find also had lower neural network complexity compared to the objective-based approach. There are still issues that were discovered with novelty search however where it wasn't always able to find high-fitness solutions. This is where local competition can aid novelty search to find higher performing solutions.

2.5 Novelty Search with Local Competition

Competition between agents is a powerful technique that is often used in artificial intelligence. A popular example of this is using competition between agents in a game of hide and seek [1]. The agents were grouped into teams, one team was hiding and the other team was seeking. The algorithm was run for millions of iterations, each time the agents learning ways to outperform the other team. The results were very interesting and the agents managed to come up with unexpected and clever solutions to beat the other team. This theory of competition could be a powerful technique to use in the field of swarm robotics.

An experiment was done to test this [11]. The goal of the experiment was to evolve locomoting virtual creatures to create a wide diversity of well-adapted creatures. Four tests were done, and the results compared. The tests done were an objective-based solution, novelty search solution, novelty search with global competition and novelty search with local competition. The difference between global and local competition is that all agents compete in global competition whereas only agents of a similar type (similar morphology) compete against each other in local competition. The first metric compared is the maximum fitness. Novelty search with global competition was able to outperform all the other solutions for this test. Objective-based and novelty search with local competition had similar results performing slightly worse than global competition and novelty search alone performed very poorly. The second metric was the niche coverage. This explained how well morphological diversity was encouraged and maintained. Novelty search and novelty search with local competition performed the best in this test whereas objective-based and novelty search with global competition performed the worst. The final metric is niche exploitation which shows how well diverse individuals that exploit niches have been found and maintained. Novelty search with local competition outperformed the other solutions in this test with novelty search alone performing poorly and the other two solutions performing slightly worse than local competition.

A similar experiment was done looking at evolving roguelike dungeons for use in virtual environments such as video games [14]. The results compared novelty search with local competition to an object-based approach and novelty search alone. The resulting fitness values and diversity were compared. The objective-based approach was able to outperform novelty search and novelty search with local competition in the fitness value test, with novelty search and local competition slightly outperforming novelty search. Novelty search and novelty search with local competition showed similar levels of diversity and significantly outperformed the objective-based solution.

These experiments show that novelty search with local competition expands on novelty search and is shown to provide the benefit of obtaining a wide diversity of solutions whilst finding the best solutions for each niche. It may not always provide the highest global fitness values that an objective-based approach can provide but provides a good balance between diversity and fitness. This is ideal for our hybrid method to improve on the deficiencies within the mEDEA method.

2.6 Summary

We looked at mEDEA algorithm as a baseline as it is able to adapt well to unknown or changing environments. There have also been various improvements to this algorithm with the study on integrating task-driven selection to mEDEA [6] and the study using various embodied distributed quality diversity algorithms which are a variation of the mEDEA algorithm [9]. Whilst mEDEA provides good environment adaptation it may not necessarily provide the highest fitness values. Novelty search and novelty search with local competition is able to improve on this and provide solutions that are more diverse while still having high fitness values [5, 11]. There are still cases though where novelty search and

novelty search with local competition have lower fitness than an objective-based solution [5, 14].

It is clear from all the studies that the results seemed to vary depending on the problem. There is not one solution that fits all problems and outperforms all other solutions in every aspect. This is something that still needs to be researched further and improved. Novelty search and local competition offers a good overall solution however it may not always be able to outperform the fitness values of certain objective-based searches. Novelty search and local competition in combination with the mEDEA algorithm to provide adaptability to unknown or changing environments should be a good method for a wide variety of problems. Further research needs to go into finding new or hybridized methods than can further improve the performance and diversity of swarm robotics solutions as well as deal with more complex tasks.

3 DESIGN & IMPLEMENTATION

The following section contains the design and implementation for the mEDEA method and our hybridized novelty search and local competition (NSLC) method as well as the heuristics, neural network and various metrics used.

3.1 Robot Controller

The robot controller is responsible for the movement and functionality of the robots. The robots are circular and have eight sensors spread equally around their body to measure the distance to objects. The values from the sensors are used to get the inputs that are fed to the neural network, section 3.2. The robots also have wheels to drive the robot around. The rotation and translation values of each robot can be changed, and the controller will then calculate what speed each wheel needs to be driven at to achieve this. Each robot will have their own robot controller to control them. We have implemented two separate robot controllers, one for the mEDEA method and one for the hybridized novelty search and local competition method. Initially both controllers are assigned random weights which correspond to the genome and will be assigned to the neural network (section 3.2) to get the output values that drive the robot. Every iteration the objects around the robot are checked to see if the pushing heuristic (section 3.3.1) or collision avoidance heuristic (section 3.3.2) needs to be applied. If neither heuristic is applied the robot will be moved based on the output from the neural network.

3.3.1 mEDEA Controller Specifics

Each iteration the mEDEA controller will share its current weight values with other robots around itself as well as receive the weight values from other robots. These weight values are stored in a received weights list. Every 1000 iterations a new generation happens. For each new generation the mEDEA controller randomly selects new weights from the received weights list. These new weights are then randomly mutated using a normal distribution and set as the current weights for determining the robot's output. The received weights list is also cleared. If at the beginning of a new generation the robot has not received any weights from other robots, that robot will be deactivated for the next generation. This means the robot will not be able to move around but can still receive weights from other robots.

3.3.2 NSLC Controller Specifics

The NSLC controller is slightly more complicated than the mEDEA controller. The weights (genome) are stored in combination with the novelty metric and fitness value for that genome in what we have called an archive item. The details of what the novelty metric and fitness value contain can be seen in section 3.4. The NSLC controller contains an archive which is a list that stores all the archive items that contains unique behaviours based on the novelty metric and local competition score. Each iteration the controller will share its archive with nearby robots and receive archives from other robots. These archives are stored in a received archives list. Every 1000 iterations there is a new generation. The novelty metric for the just run generation is first calculated. This novelty metric is used to calculate the average distance to the nearest k neighbours in the archive based on their novelty metric's using the following equation:

$$p(x) = \frac{1}{k}\sum_{i=0}^{k} dist(x, u_i),$$

Where x is the current novelty metric we are comparing and u_i is the ith-nearest neighbour with respect to x. This average distance is the novelty search part of the implementation. The fitness value is also compared to the fitness values of the same nearest k neighbours in the archive. If the fitness value is greater than that of a neighbour, the local competition score is incremented. This local competition score is the local competition part of the implementation. The average distance and local competition score are then used in a weighted sum to calculate a threshold value. If this threshold is greater than some value n, the genome, novelty metric and fitness values from the generation are added to the archive. A new genome must then be selected for the next generation. To do this the received archive list is flattened and a random archive item is selected from this list. The weights are then extracted from this archive item and mutated in the same way as the mEDEA controller. These weights then become the active genome for the next generation. And the received archive list is cleared. As with the mEDEA controller if a robot has not received any archives, it will be deactivated and remain stationary for the next generation but will still be able to receive archives.

3.2 Neural Network

A neural network is used in order take the inputs from the sensors and provide two outputs to drive the wheels of the robots. The neural network used is a multilayer perceptron neural network. This is a fully connected, feed forward artificial neural network [22]. The inputs provided to the neural network are the distances to the

closest object from each of the eight sensors (if no object the value is 1). In addition to the distances, whether that object is a robot, a wall or a resource is also provided as inputs. The neural network can also be set up to have hidden layers. The weight values for all the connections come from the current active genome of the robot. Each iteration the inputs and weights will be loaded into the neural network. It will then step through the neural network using a tanh activation function to calculate two output values between -1 and 1 to drive the translation and rotation speeds of the robot's wheels.

3.3 Heuristics

Two heuristics were designed and implemented in order to handle pushing a resource into the target area as well as collision avoidance.

3.3.1 Pushing Heuristic

The pushing heuristic is used to allow a robot to collect and push a resource into the target area. There are three different resources that can be pushed into the target area, small, medium, and large. Small resources only require one robot to push them. Medium resources require two robots and large resources require three robots. A robot will follow this heuristic once it is a certain distance away from a resource that has not yet been pushed into the target area and has less than the required number of robots currently attached to it (Is not already being pushed to the target area by other robots). Once a robot is following the pushing heuristic it will bind itself to the resource. If there are less than the required number of robots bound to the resource in order to push the resource, the robot will wait for other robots to come help. If, after a set number of iterations of waiting, no robots come to help push the resource, the robot will unbind itself from that resource, stop following the pushing heuristic and continue exploring based on the usual outputs from the neural network. The same will happen if the resource gets stuck somewhere for a certain number of iterations while being pushed to the target area. Once the required number of robots have attached to the resource, they will push the resource to a random location in the target area using the angle to the target location as well as the robot and resource's position. Once the resource has been pushed to the target location the robots will unbind from the resource and continue exploring.

3.3.2 Collision Avoidance Heuristic

The collision avoidance heuristic will take over when a robot is going to collide with another robot or when a robot is going to collide with a resource that is either already in the gathering zone or is currently being pushed by the required number of robots to the gathering zone. The collision avoidance heuristic will steer the robot away from any collisions and once clear the robot will then continue to explore from the outputs of the neural network.

3.4 Novelty Metric

The novelty metric is used in the novelty search implementation to determine how different certain behaviours are from each other. In our implementation two values were used to make up the novelty metric. The first value is based off how many resources a robot pushed into the gathering zone during a generation. If it is a small resource the robot gets one point and medium and large resources get two and three points respectively. This value is also our fitness metric. The second value used is the total distance a robot travels during a generation. This value is not simply the difference between the robot's position at the start and end of the generation but rather the total distance driven by the robot throughout the generation. These two values are stored in an array to create the novelty metric. The Euclidean distance between two novelty metric's is then used in our hybridized method to determine if behaviours are different from each other and whether to add it to the archive.

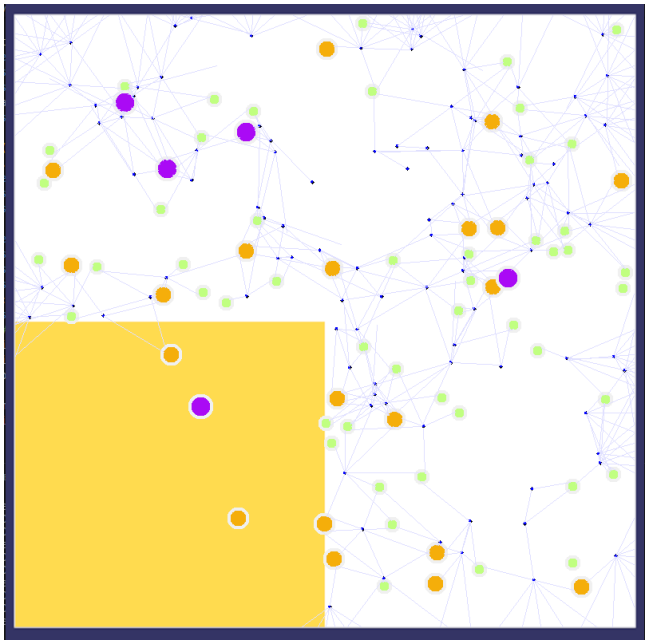

Figure 2: Example of collective gathering task environment

3.5 Evaluation Metrics

In order to compare mEDEA with the hybridized method certain metrics needed to be used. We compared both the performance and diversity of the methods. These metrics are then used as described in section 4 to obtain the results.

3.5.1 Performance Metrics

Four metrics are used to determine the performance of both of the methods:

- Total number of resources pushed into the gathering zone during a run
- Time taken to complete the task (push all resources into the gathering zone) as a percent of the total running time
- Maximum fitness value of all the genomes from a run
- Average fitness value of all the genomes from a run

The fitness value is defined in section 3.4.

3.5.2 Diversity Metrics

Four metrics are used to determine the diversity of both of the methods:

- Average swarm distance to the center of the gathering zone for each generation throughout a run
- Average time (in iterations) spent pushing a small resource for each generation throughout a run
- Average time (in iterations) spent pushing a medium resource for each generation throughout a run
- Average time (in iterations) spent pushing a large resource for each generation throughout a run

4 EXPERIMENT DESIGN

To compare the mEDEA method with our hybridized method experiments were setup with a collective gathering task. A collective gathering task is a simple, commonly used task in swarm robotics. The task works by populating the environment with resources. A certain portion of the environment will be designated as the gathering zone. It is the goal of the robots to push all the resources into the gathering zone. An example of this task environment is shown in figure 2 and explained in more detail later in this section. This task is suitable for comparing our methods as it is a simple task and is easy to measure the task performance and diversity of the solutions.

Three different resource types are used for the task, specified as small, medium, and large resources. Each type of resource has a different size and requires a different number of robots to push the resource as shown in table 2. There are then three different task environments used in the experiments, specified as easy, medium, and hard. Each task environment has a different proportion of small, medium, and large resources as shown in table 2. The total resources remain constant for all the task environments at 75. An example of a task environment can be seen in figure 2. The small blue dots each represent a robot and the lines coming out of the robot represent a sensor that currently can see an object. The green, orange, and purple circles represent the small, medium, and large resources respectively. Finally, the yellow area at the bottom left of the arena represents the gathering zone where the robots need to push the resources into.

Table 1: Simulation Parameters

Number of Robots	100
Number of Resources	75
Arena Width x Height	800x800px
Gathering Zone Width x Height	400x400px
Time steps per trial	10000
Generations per trial	10
Iterations per generation (Robot Lifetime)	1000
Number of Experiment runs (for Each Task Environment)	20

Table 2: Experiment Parameters

Resource Radius	Small	6
	Medium	10
	Large	12
Task Environments	Easy	
	Medium	
	Hard	
Number Blocks for Each Task Environment (small, medium, large)	Easy	50, 20, 5
	Medium	25, 30, 20
	Hard	15, 25, 35
Robots Needed to Push Resource	Small	1
	Medium	2
	Large	3

The performance and diversity metrics mentioned in section 3.5 are used and averaged for 20 runs for each of the two methods in combination with each of the three task environments (six combinations in total). An additional performance metric which looked at the maximum fitness of all of the genomes over all of the runs for each combination of method and task environment was also added. The performance metrics were graphed using a bar graph and an independent t-test was done to compare the differences in the two methods. The diversity metrics were graphed using a scatter plot to be able to visible see the difference in diversity between the two methods for each generation as well as bar graphs averaged over an entire run.

5 RESULTS & DISCUSSION

The results for the comparison of the performance and diversity of the two methods are shown and discussed in this section. Only the most relevant graphs and results have been shown in this paper. The rest of the results and graphs can be seen in the GitHub repo, link in appendix A

5.1 Task Performance Results

As seen in figure 3 which compares the average maximum number of resources pushed into the gathering zone as a portion of the total number of resources for each method and task environment, mEDEA and NSLC performed very similarly. As the task environment difficulty level increases the portion of total resources pushed into the gathering zone decreases equally for both mEDEA and NSLC. This decrease is to be expected due to the higher complexity however both methods still perform fairly optimally, even at the hardest difficulty, with the lowest average portion being 84%. It is likely that given enough time that these portions will be much closer to 100% for both methods, even for the hardest difficulty. As shown in table 4 no significant difference is seen between mEDEA and NSLC for the average maximum resources pushed into the gathering zone for all difficulty levels. While it may appear there is a difference in the hard task environment there was still not enough evidence to show any differences. This is based off an independent t-test that was done and a significance level of p=0.05 is used. This test compares the mean values for the two methods for the particular metric assuming unequal variances to judge if there is a significant difference. The exact p-values from the test can be seen in the GitHub repo in appendix A.

Figure 4 shows the results of the average time taken to complete the task as a portion of the total simulation time for each method and task environment. Completion of the task entails all the resources being pushed into the gathering zone and so if at the end of the simulation runtime not all the resources have been pushed into the gathering zone, this value will be 100%. As with the average maximum resources pushed, mEDEA and NSLC performed very similarly in this test as well. The average time taken increased as the task complexity became harder, which is to be expected. The average time taken was 100% for both methods for the hardest difficulty, indicating there were no runs in which all the resources were pushed in the gathering zone. Again as seen in table 4, there are no significant differences between mEDEA and NSLC for the average time taken to complete the task.

There are more metrics that are compared and shown in the GitHub repo in appendix A. They all show a similarity in performance between mEDEA and NSLC. The t-tests showed no significant difference for any of the performance metrics that were recorded. We will further discuss why this could be the case in section 5.3.

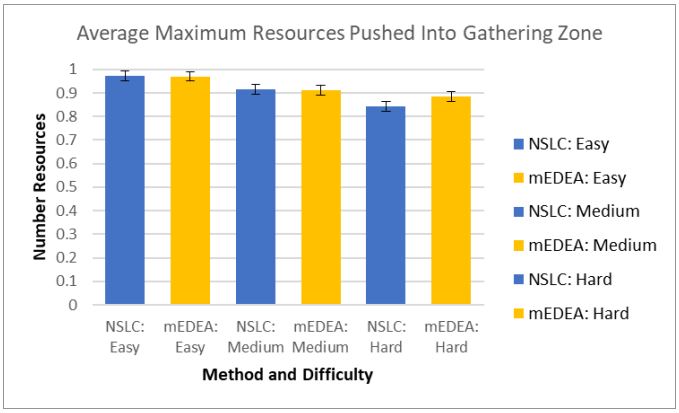

Figure 3: Bar graph showing the average maximum number of resources pushed into the gathering zone as a portion of the total number of resources for each method and task environment

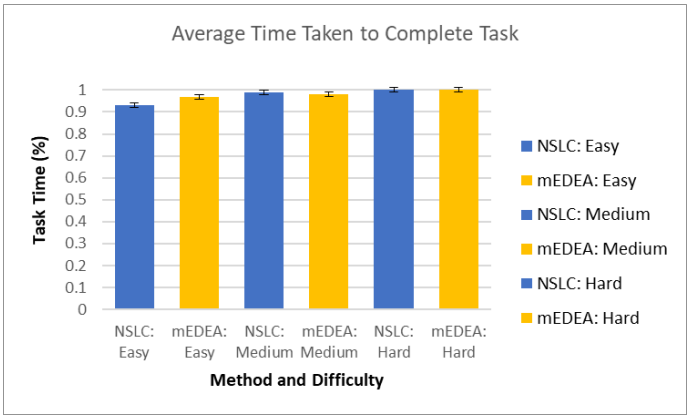

Figure 4: Bar graph showing the time taken to complete the task as a portion of the total simulation time for each method and task environment

	Average Maximum Resources Pushed into Gathering Zone	Average Time Taken to Complete Task
Easy Difficulty	No Significant Difference	No Significant Difference
Medium Difficulty	No Significant Difference	No Significant Difference
Hard Difficulty	No Significant Difference	No Significant Difference

Table 4: Showing if there is a significant difference between the NSLC and mEDEA for various metrics at each difficulty level based on an independent t-test

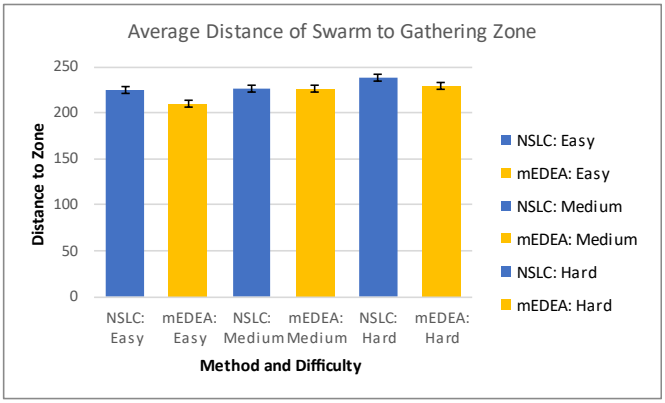

Figure 5: Bar graph showing the average distance of the swarm to the center of the gathering zone for each method and task environment to measure behavioural diversity of the swarm

	Average Distance of Swarm to Gathering Zone
Easy Difficulty	Significant Difference
Medium Difficulty	No Significant Difference
Hard Difficulty	Significant Difference

Table 3: Showing if there is a significant difference between NSLC and mEDEA for average distance to swarm at each difficulty level based on an independent t-test

5.2 Diversity Results

The behavioural diversity of the solutions is slightly more challenging to compare than the performance. The behavioural diversity looks for a significant difference in the behaviours of the robots and we defined how this is calculated in section 3.4. Figure 5 shows a bar graph of the average distance of the swarm to the center of the gathering zone over the course of a run for each method and task environment. It is seen that as the difficulty increases the average swarm distance tends to increase. This is most likely due to the higher level of cooperation needed in the more complex tasks requiring robots to wait longer and spend more time exploring away from the gathering zone. As shown in table 3, an independent t-test was done to see if there were any significant differences in the average swarm distance from the gathering zone for each method. A significant difference was found for the easy and hard task environments but not for the medium. The details of each task environment is defined in section 4.

While a difference was found in the easy and hard environments, care must still be taken in interpreting what this means for the diversity of the solutions. In the easy and hard environments the NSLC method had a greater average distance from the gathering zone than mEDEA. This could mean the NSLC method was able to explore further from the gathering zone and could be a possible indication of more diverse solutions. This was not apparent in the medium difficulty level however and the significant difference could be due to other factors such as the randomized starting locations of the resources or the NSLC method spending a longer time waiting for cooperation than the mEDEA method. The other diversity metrics that were captured, which can be seen in the GitHub repo in appendix A, also show a similar trend between the NSLC and mEDEA methods. There is therefore not enough evidence to safely judge whether there is a significant difference between the diversity of the solutions from the two methods. We will go into more detail discussing why this could be the case in section 5.3.

5.3 Discussion

Based on previous studies with novelty search and novelty search with local competition [5, 11], we were expecting our hybridized NSLC method to be better performing and more diverse than the base mEDEA method. From our experiments we were not able to find any significant differences between the two methods. There are a few possible reasons why this could be the case.

The collective gathering task used to compare these methods is quite a simple task and so it was seen that even the base mEDEA method was performing optimally (with a lowest maximum average of 89% of the resources pushed into the gathering zone for the hardest task environment, shown in figure 3). This means that there is not much room for an improvement in performance for our hybrid method for this particular task.

Another factor that could affect the diversity and performance is the pushing and collision avoidance heuristics. These heuristics will be active for a large portion of time while a robot is pushing a resource to the gathering zone or avoiding a collision. While these heuristics are active the mEDEA or NSLC methods has nothing to do with the movement of the robot. These two heuristics are also identical for both methods. This would mean that given a random uniform starting location of the resources, there should be no difference in the portion of time spent pushing a particular type of resource to the gathering zone for either method and once a resource has been found it should have an equal chance of being successfully pushed to the gathering zone for both methods. It also means that the mEDEA and NSLC methods are only responsible for exploration and finding resources. Exploration and environment adaptation is one of the main strengths of the mEDEA algorithm [2] so it is no surprise that it was able to perform well in this regard and since our hybrid method uses mEDEA as a base it would have adopted this strength.

There was a significant difference in the average distance of the swarm to the gathering zone for the easy and hard task environments (table 3). While this could indicate NSLC showing more exploration among robots and a higher diversity, as seen in the previously looked at studies [11, 14], there is still not enough evidence to properly conclude whether this is the case.

We have found with a simple collective gathering task our hybrid method does not provide any significant performance or diversity improvements over the base mEDEA method (table 3 and table 4). We still believe our hybrid NSLC method has potential in more complex problems where the weaknesses of mEDEA [2] may be more of a problem and the strengths of novelty search and local competition to have high diversity while still having high performing solutions [11, 14] will be more apparent. We propose further research into comparing our hybrid to mEDEA in more complex tasks such as categorization [15], aggregation [19] and group transport [17] to see if this is the case.

6 CONCLUSION

This study aimed to determine the best method for evolving problem-solving collective behaviour and functional diversity in swarm robotics. Two methods were tested. The first method was the mEDEA algorithm, an environment-driven adaptation method [2]. The second method was our hybridized method using mEDEA as a base and combining it with novelty search with local competition [11]. The mEDEA method has been shown to be able to provide good environment adaptation but is lacking in other important areas [2]. Our hybridized method aimed to combine the environment adaptation strengths of the mEDEA algorithm and the high diversity and high-performance strengths of novelty search with local competition. The expectation was that our hybridized method would be better performing and have a higher diversity then the mEDEA method.

The two methods were compared in a simple collective gathering task where the swarm of robots are required to push resources into a set area. The methods were tested at different levels of complexity where different sized resources were used. Three different sizes were used, small, medium and large each requiring one, two and three robots to push respectively. Using different ratios of these resources three different task environments were used in the experimentation, each with a different level of complexity. The performance and diversity of these results were then compared. Upon analyzing the results no significant difference could be found between the performance and diversity of the two methods. There was a difference in the average distance of the swarm to the gathering zone in the easy and hard task environments however, with the NSLC method having a higher average distance. This could be an indication that NSLC was able to explore further away from the gathering zone than mEDEA. It could also be due to other reasons however such as having to wait longer for cooperation. These are not the results we were expecting but discussed a few reasons why there may not have been a significant difference between the two methods in section 5.3.

While our hybridized method showed no difference to the base mEDEA method for this particular task, it still performed to an equal high standard as the mEDEA method. We still believe our hybridized method has the potential to outperform the mEDEA method in more complex tasks in which the strengths of novelty search and local competition will shine through. This is where we think further research needs to be done for determining the best method for evolving problem-solving collective behaviour and functional diversity in swarm robotics.

REFERENCES

[1] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent Tool Use From Multi-Agent Autocurricula. (2019). DOI:https://doi.org/10.48550/arXiv.1909.07528

[2] Nicolas Bredeche and Jean-Marc Montanier. 2010. Environment-driven Embodied Evolution in a Population of Autonomous Agents. "Parallel Problem Solving From Nature (2010).

[3] Stephane Doncieux, Alban Laflaquière, and Alexandre Coninx. 2019. Novelty search: a theoretical perspective. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19). Association for Computing Machinery, New York, NY, USA, 99–106. DOI:https://doi.org/10.1145/3321707.3321752

[4] Gusz Eiben and James E. Smith. 2015. Introduction to Evolutionary Computing (2nd ed.). Springer Berlin.

[5] Jorge Gomes, Paulo Urbano, and Anders Lyhne Christensen. 2013. Evolution of swarm robotics systems with novelty search. Swarm Intelligence 7, 2–3 (2013), 115–144. DOI:https://doi.org/10.1007/s11721-013-0081-z

[6] Evert Haasdijk, Nicolas Bredeche, and A. E. Eiben. 2014. Combining Environment-Driven Adaptation and Task-Driven Optimisation in Evolutionary Robotics. PLoS ONE 9, 6 (2014), e98466. DOI:https://doi.org/10.1371/journal.pone.0098466

[7] Heiko Hamann. 2018. Swarm Robotics: A Formal Approach (1st ed.). Springer Cham.

[8] Aboul Ella Hassanien and Eid Emary. 2016. Swarm Intelligence: Principles, Advances, and Applications (1st ed.). CRC Press.

[9] Emma Hart, Andreas S. W. Steyven, and Ben Paechter. 2018. Evolution of a functionally diverse swarm via a novel decentralised quality-diversity algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18). Association for Computing Machinery, New York, NY, USA, 101–108. DOI:https://doi.org/10.1145/3205455.3205481

[10] Joel Lehman and Kenneth O. Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. Evol. Comput. 19, 2 (Summer 2011), 189–223. DOI:https://doi.org/10.1162/evco_a_00025

[11] Joel Lehman and Kenneth O. Stanley. 2011. Evolving a diversity of virtual creatures through novelty search and local competition. In Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO '11). Association for Computing Machinery, New York, NY, USA, 211–218. DOI:https://doi.org/10.1145/2001576.2001606

[12] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. 2016. Quality Diversity: A New Frontier for Evolutionary

Computation. Frontiers in Robotics and AI 3, (2016), 40. DOI:https://doi.org/10.3389/frobt.2016.00040

[13] Adam Slowik and Halina Kwasnicka. 2018. Nature Inspired Methods and Their Industry Applications ―Swarm Intelligence Algorithms. IEEE Transactions on Industrial Informatics 14, 3 (2018), 1004–1015. DOI:https://doi.org/10.1109/tii.2017.2786782

[14] Alexandre S. Melotti and Carlos de Moraes. 2019. Evolving Roguelike Dungeons With Deluged Novelty Search Local Competition. IEEE Transactions on Games. 11, 2 (2019), 173-182. DOI:https://doi.org/10.1109/TG.2018.2859424

[15] Christos Ampatzis, Elio Tuci, Vito Trianni, and Marco Dorigo. 2008. Evolution of Signaling in a Multi-Robot System: Categorization and Communication. Adaptive Behavior 16, 1 (2008), 5–26. DOI:https://doi.org/10.1177/1059712307087282

[16] Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E. (Gusz) Eiben. 2015. Evolutionary Robotics: What, Why, and Where to. Frontiers in Robotics and AI 2, (2015). DOI:https://doi.org/10.3389/frobt.2015.00004

[17] Roderich Groβ and Marco Dorigo. 2008. Evolution of Solitary and Group Transport Behaviors for Autonomous Robots Capable of Self-Assembling. Adaptive Behavior 16, 5 (2008), 285–305. DOI:https://doi.org/10.1177/1059712308090537

[18] Wenguo Liu, Alan F T Winfield, and Jin Sa. 2007. Modelling swarm robotic systems: A case study in collective foraging. Towards autonomous robotic systems (TAROS 07) 23, (2007), 25–32.

[19] Vito Trianni, Roderich Groβ, Thomas H. Labella, Erol Sahin, and Marco Dorigo. 2003. Evolving Aggregation Behaviors in a Swarm of Robots. Advances in Artificial Life (2003), 865–874. DOI:https://doi.org/10.1007/978-3-540-39432-7_93

[20] Vito Trianni, Stefano Nolfi, and Marco Dorigo. 2006. Cooperative hole avoidance in a swarm-bot. Robotics and Autonomous Systems 54, 2 (2006), 97–103. DOI:https://doi.org/10.1016/j.robot.2005.09.018

[21] Jagdish Chand Bansal, Pramod Kumar Singh, and Nikhil R. Pal. 2019. Evolutionary and Swarm Intelligence Algorithms (1st ed.). Springer Cham.

[22] Ke-Lin Du and M. N. S. Swamy. 2019. Neural Networks and Statistical Learning (2nd ed.). Springer London.

APPENDIX A

GitHub repo containing results, source code and documentation. Note the results are in csv and excel formats with performance.xlsx and diversity.xlsx containing the graphs and analysis and the csv files containing the data from the experiment runs: https://github.com/BaileyGreen/NSLC