Swarm Robotics: Novelty Search and Local Competition

Literature Review

Bailey Green
University of Cape Town
GRNBAI001@myuct.ac.za

ABSTRACT

Swarm robotics is a complicated and constantly changing field. There exists many methods and algorithms to deal with problems, each of these methods having their own advantages and disadvantages. This review starts by looking an introduction to swarm intelligence. Two specific nature inspired methods are shown to give an example of what swarm intelligence is about and then a quick look into evolutionary algorithms and reinforcement learning. We then look at swarm robotics which is an application of swarm intelligence and the main focus of this review. An overview of evolutionary robotics is given which is a field within evolutionary computing and many of the methods looked at in this review make use of evolutionary robotics to implement a swarm robotic solution. We then look at a few of the methods used in swarm robotics, in particular at environment adaptation, objective-based methods and behavior-based methods. A large part of the review has to do with novelty search and its variations and using this in combination with local competition. The review ends with a discussion comparing the methods looked at and mentioning where there could still be further improvement in the field.

1. INTRODUCTION

Swarm intelligence follows the behavior 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 [15]. Two common nature inspired swarm intelligence methods are particle swarm optimisation (PSO) and ant colony optimisation (ACO).

Particle swarm optimisation is an optimisation method that simulates the behaviors of animals. Particle swarm optimisation is also inspired by evolutionary algorithms. The algorithm consists of a swarm of particles where each particle is a solution determined by a fitness function. Initially the particles are randomly generated. Each particle then moves in a direction that depends on its current movement, its personal best, and the global best values. These values then get updated after each iteration [23]. One of the areas in which particle swarm optimisation is used widely is electric power systems [6]. It was mentioned the reason for using particle swarm optimisation is due to three main reasons. The first is that it is easy to implement and has fewer parameters for tuning. The second is due to its more efficient memory capabilities. And lastly is due to its efficiency in maintaining the diversity of the swarm.

Algorithm 1 Genetic Algorithm

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 genetic operators (crossover, mutation) to generate offsprings
    Replace a proportion or the entire population with the offsprings
End While

Figure 1: Showing pseudo code for a typical evolutionary algorithm [27]

Ant colony optimisation is a technique that is inspired by the swarm intelligence of ant colonies. When ants search for food they initially start by exploring a random area around the nest. While these ants are exploring, they leave behind a pheromone chemical trail. This allows the ants to efficiently search for food by using concentrations of the pheromone chemical to best choose their paths [23]. Ant colony optimisation uses this theory to create an algorithm that can be used in many real-world applications such as cloud computing [9] and feature selection [25].

Evolutionary Algorithms is another technique inspired from natural evolution and Darwin's survival of the fittest strategy [27]. It is the baseline principles for some of the methods and techniques that will be described later in this review. It works by using the foundations of evolution and DNA. As shown by the pseudo code in figure 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 than 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. Simple evolutionary algorithms are often limited by slow convergence properties and limited diversity. More advanced methods are required to overcome this limitation.

While not a direct application of swarm intelligence or evolutionary algorithms, reinforcement learning is another technique that can be used to provide adaptable and efficient solutions to problems. Reinforcement learning is a type of machine learning where agents are constantly learning the most rewarding actions to take [35]. It shared a lot of the same goals and similarities with evolutionary algorithms. The agents are not told what to do but must discover which actions yield the most reward by trying them. An important

part of reinforcement learning is the trade-off between exploration and exploitation. Exploration involves the agent making random actions or decisions in order to further explore an environment and its associated rewards. Exploitation involves choosing the action that will maximise its reward. Its an important trade-off as not enough exploration will lead to an agent potentially not finding the most efficient solution or any solution at all. On the other hand too much exploration and the agent will spend its whole time exploring and never following the best solutions. A common solution to this trade-off is to start by allowing the agent to explore and each iteration slowly make the agent favour exploitation more and more. Reinforcement learning is a common and powerful technique for solving complex problems and can be used in the field of robotics. We have mentioned it in this review due to its similarities with evolutionary algorithms as well as the potential to use it in combination with some of the methods mentioned in this review.

Swarm robotics is a field in swarm intelligence and what we will be focusing on in this review. Swarm robotics is a research field in which a group of robots work together in order to perform some task. 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 be either 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 for example. These robots can either be physical robots interacting with real world environments or can be simulated robots interacting with a setup virtual environment. There are various techniques and algorithms available in order to achieve swarm intelligence in swarm robotics. These various techniques each have their own strengths and weaknesses and can be used to create a swarm of robots that can solve complex tasks in an adaptable and scalable way in changing environments [22, 24].

2. Evolutionary Robotics

Evolutionary robotics is a field that uses evolutionary computing and evolutionary algorithms to solve tasks using robotics [29]. 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. Common swarm robotics tasks solved using an evolutionary approach include foraging [31], aggregation [32], hole avoidance [33], categorization [28] and group transport [30].

Evolutionary robotics has been widely researched and there a few key things that are commonly accepted in the field [29]. 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. This issue will be explored later in this review but can be solved by using a behaviour-based approach rather than an objective based approach. 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. For this reason, this review will mainly be focusing on swarm robotics systems and methods in simulated environments.

Due to the strengths and adaptability of evolutionary robotics we will be looking at methods that make use of evolutionary robotics for the use in solving tasks using swarm robotics in this review.

3. Overview: Swarm Robotics Methods

Solving complex problems using swarm robotics is a difficult task and something we have not mastered yet. There is still lots of ongoing research in this field. Depending on the problem you are trying to solve, there may be different solutions that best fit that problem. The three big areas when looking at methods for implementing swarm robotic systems are, being adaptable to unknown or changing environments, being able to achieve a high max fitness value to be able to solve the task effectively and having a diverse swarm. There are many methods each with their own strengths and weaknesses which can be used to achieve these goals.

3.1 Environment Adaptation

Being able to adapt or react to unknown environments is important for swarm robotic systems. Environments in the real world are unpredictable and cannot be properly controlled unlike virtual environments. There are many different ways to achieve this. A simple way that can be applied to most algorithms is to balance the exploration versus exploitation the agents are doing [3]. Exploration has to do with visiting entirely new areas of the environment or search space whereas exploitation is taking advantage of the areas you have already visited. Balancing these two factors is important in allowing the agents to be able to explore the environment while still finding effective solutions. This solution is simple and rather easy to implement but will not help much with complex and changing environments. More complex environments will require inspiration from swarm intelligence to solve. A study was done in which a modified artificial bee colony was used in combination with evolutionary computation to try and solve a navigation task in an unknown environment [10]. The results show the algorithm was able to cope well with a complex and unknown environment. The robots were able to consistently find a feasible path through the environment with minimal collisions. While this solution has proven to work well with a navigation task, other problems may require different solutions. Environment-driven distributed evolutionary adaptation (EDEA) is a more common and well-known solution to unknown and changing environments [2]. We will further explore EDEA later in this review due to its adaptability and ability to provide solutions to deal with these unknown environments.

3.2 Objective-based Methods

The fitness function is used in evolutionary robotics to determine which agents to select and reproduce. It measures how close to a given solution these agents are. The higher the fitness of an agent the better the solution to the problem [16]. It makes sense then to want to find methods to maximise the fitness function. These methods are often referred to as objective-based methods. They are implemented in a very similar way to evolutionary algorithms, as described earlier in the introduction and as such face the same

limitations. One of the common issues with objective-based search is getting stuck in a local maximum. This is where the algorithm will too quickly converge on a particular solution that may be good but is not the overall best solution. This happens when there is not much diversity in the solutions. A way to solve this is to use a behavior-based search rather than an objective-based search.

3.3 Behavior-based Methods

Behavior-based search uses the behaviors of the robots in a swarm as a control rather than an objective. A behavior could simply be obstacle avoidance or exploration. Behavior search often involves robots communicating with each other and sharing information. A study was done using a behavior-based control method in a virtual search and rescue task [7]. The results show the swarm behavior-based control method was able to identify victims with a higher percentage over a single-robot, multi-robot and APO search methods. It also showed that the swarm behavior-based search covered around 1.5 to 2 times the area compared to the other methods. There are many behaviour-based methods that exist and have been previously researched.

Quality diversity algorithms are a type of behaviour-based method. There are many different quality diversity algorithms, but they all share the same outline and goal [21, 34]. The goal of quality diversity algorithms is to develop as many behavioural niches as possible but have each niche represented by the highest possible fitness solutions. A key concept in all quality diversity algorithms is tracking the behaviours of an individual. The behaviours can then be compared to other individuals which drives the diversity component of the algorithm. These behaviours can also be aligned with some fitness value which drives the quality component of the algorithm.

The MAP-Elites algorithm is one these quality diversity algorithms that use this behaviour-based search [20]. The algorithm works by using the fitness function to measure some performance behaviour metric (i.e., Speed of a robot). A feature space of interests is also defined. This could contain information such as height and weight of robots for robot morphologies. MAP-Elites then searches for the best solution for each interest in the feature space. A study was done using MAP-elites algorithm in three different search spaces: neural networks, simulated soft robot morphologies, and a real, soft robotic arm [20]. As expected, the results showed MAP-Elites being able to find higher performing solutions than many of the various control algorithms it was compared to.

Another popular quality diversity algorithm is novelty search. Novelty search is useful for finding a wide diversity of solutions and avoiding the problem of converging to a local maximum. It works by looking at the behaviour of agents instead of some objective. We will be looking at novelty search and some of its variations in more detail later in this review, section 5.

Algorithm 1 The MEDEA algorithm

```
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
```

Figure 2: mEDEA algorithm [2]

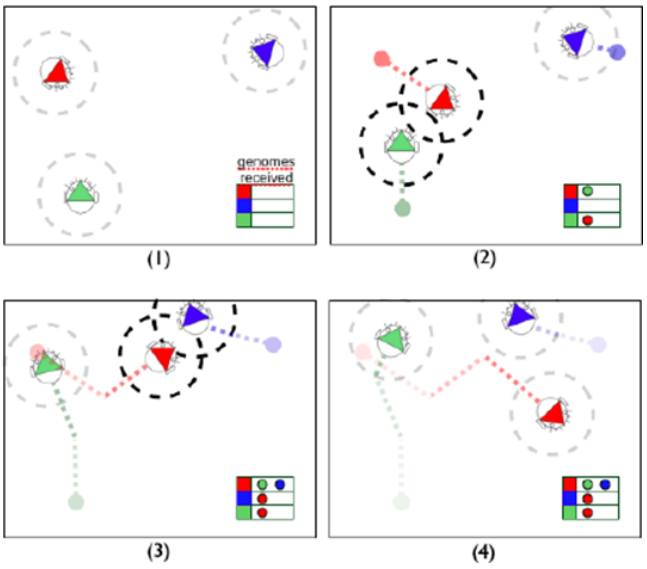

Figure 3: Showing agents broadcasting and receiving genome information from other agents in range [2]

4. mEDEA

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 to ensure maximum genome sharing as well as environment interactions [2].

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

shown in figure 3. At the end of each generation the current genome is emptied. There are then three operators that are then 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].

There are various adaptions and improvements people have made to the mEDEA algorithm in order to improve its efficiency and suitability to certain tasks. mEDEA algorithm works well for reacting to unknown or changing environments but not as much for task-based evolution.

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 [13]. Another study done comparing the mEDEA algorithm to various embodied distributed quality diversity algorithms (EDQD) which are a variation of the mEDEA algorithm [14]. 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.

5. 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 [17]. There are two main motivations for using novelty search [8]. The first being obtaining a particular goal space as quickly and frequently as possible [17]. The second motivation for using novelty search relates to quality diverse where we want to generate solutions that are as diverse as possible [21].

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

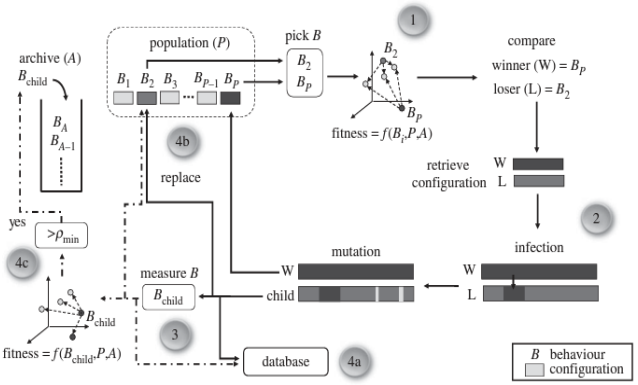

Figure 4: Novelty Search Implementation [5]

in the archive (above some threshold) that behavior is added to the archive [11]. Figure 4 below shows an implementation of the novelty search taken from a study on an experimental framework to characterise reservoir computers [5]. The framework uses novelty search in combination with a microbial genetic algorithm but figure 4 still helps to explain the basic process behind implementing novelty search.

In this particular implementation, initially a random population is created and added to archive A and population P. At step 1 a random individual is selected, and another picked based on its proximity to the first. The novelty (fitness) of both individuals are calculated relative to the population and archive. The individual with the larger fitness value is then set as the winner. At step 2 the configurations of both individuals' behaviours are retrieved. The weaker individual then undergoes gene transfer, becoming a percentage of the winner and loser. The individual is then mutated to become the child. At step 3 the fitness of the new child is measured. At step 4a the behaviour and configuration of the child is added to the database and at step 4b the loser is replaced by the child in the population. Lastly at step 4c the fitness of the child is compared to the population and archive. If the fitness of the child exceeds some threshold the behaviour of the child is added to the archive.

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 [11]. 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. Variations of novelty search can be used in order to overcome some of these issues.

5.1 Novelty Search Variations

Minimal criteria novel search (MCNS) is an extension of novelty search. In minimal criteria novel search an individual must meet some minimum criteria before being able to be selected for reproduction. Individuals that do not meet this minimum criteria receives a novelty score of zero and a dummy value is assigned for the behavioral characterisation. If the minimum criteria is met novelty search continues as usual. Two experiments were done comparing objective-based search, novelty search and minimal criteria novel search [18]. The first was a maze where a robot had to navigate from the start point to the end point in a fixed time period. The second expanded on the maze task where the robot had to reach two different end points. The results showed that for both experiments MCNS was able to find solutions significantly more consistently than the objective-based and novelty search approaches. There are still limitations and problems that were found with MCNS. One limitation is that until an individual that meets the minimum criteria is found, the search is effectively random. The other issue is that the choice of the minimum criteria requires careful consideration and knowledge as it can significantly reduce the search space. If the search space is constrained too much it could negatively affect the evolution process.

Progressive minimal criteria novel search is an extension of minimal criteria novel search in order to try fix the limitations. It works by having the minimum criteria as a dynamic fitness threshold. The minimum criteria starts off at a low value (typically zero) and each generation a new value is calculated. This allows it to progressively restrict the search space. A resource sharing task experiment using a swarm of five homogenous robots sharing a single battery charging station was done comparing various search methods [12]. The results show that progressive minimal criteria novelty search was able to significantly outperform objective-based search and novelty search and fixed the issues associated with minimal criteria novelty search.

One of the other issues found with novelty search is that it does not scale well to large search spaces. Combining objective-based selection with novelty search using linear scalarisation is a solution to this issue. An experiment was done implementing linear scalarisation using a 6 x 6 grid world surrounded by walls [4]. Within the grid there are six blocks and the agent has eighty moves to push the blocks against the walls. The algorithm works by having a p value between 0 and 1. If $p = 0$ a pure objective-based solution is used and if $p = 1$ a pure novelty search solution is used. The experiment was run using increments of 0.1 for p, going from 0 up to 1. The results showed that values of p between 0 and 1 seemed to perform better than a pure objective-based or novelty search solution. In particular a p value of 0.8 (80% novelty search and 20% objective-based) seemed to perform best.

5.2 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 where 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 [19]. 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 [26]. 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.

6. Discussion and Conclusion

There are clearly a wide range of techniques and algorithms to solve a problem using swarm robotics. Depending on the problem you are trying to solve a different solution may better suit the task and at the moment there does not seem to be a single technique that outperforms or is more adaptable than any other.

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 [13] and the study using various embodied distributed quality diversity algorithms which are a variation of the mEDEA algorithm [14]. Whilst mEDEA provides good environment adaptation it may not necessarily provide the highest fitness values.

This is where the objective-based methods that were briefly mentioned would be used. We did not focus much on objective-based methods in this review although they were compared to novelty search and its variations in a lot of the studies that were mentioned. The reason for not focusing too much on objective-based search is because of its limitation on getting stuck in a local maximum that many objective-based approaches have, as seen in [11].

Instead, the main focus was on Novelty search and its variations which use a behavior-based method. This allows for a greater diversity and solves the issue with objective-based search. In all of the studies looked at in the standard novelty search section novelty search had varying fitness value results but was often able to match or outperform objective-based search in terms of fitness values. When looking at the MAP-Elites study however [20], MAP-Elites was able to outperform novelty search in terms of fitness values. MAP-Elites seems a technique comparable to novelty search and its variations. MAP-Elites has the benefit of mapping the entire feature space allowing the user to see the trade-offs between features and performance. One of the drawbacks found with MAP-Elites however is that it does not allow the addition of new types of cells that did not exist in the original feature space. This means MAP-Elites cannot exhibit open-ended evolution.

While no comparisons between the novelty search variations and Map-Elites could be found, the novelty search variations did show an improvement over regular novelty search. Progressive minimal criteria novelty search seemed to significantly outperform objective search and regular novelty search in the experiment done in [12]. Similarly combining novelty search and objective-based search using linear scalarisation also showed an improvement [4]. Novelty search with local competition also proved a powerful technique and showed improvements over novelty search in both the resulting fitness values as well as diversity [19]. One of the most important aspects of novelty search however that also carried over to all its variations and local competition is its ability to find a diverse set of solutions. The behaviour-based approaches of MAP-Elites, Novelty Search and the variations of Novelty Search, including Novelty Search with local competition, proved often to be better solutions than an objective-based solution.

A lot of current issues and areas where further research can be focused in swarm robotics can be seen in the open issues in evolutionary robotics [29]. One of the big issues is being able to apply solutions to real world problems. All of the studies looked at in this review have been based on common made up problems in a simulated environment. While this allows good control and is relatively inexpensive applying these problems to the real world can be challenging and is not necessarily a direct translation from the simulated world. Research could be done in applying solutions to real world problems and how to map these techniques from a virtual environment to a real world environment. Another area that could be looked at is combining evolution with learning, such as reinforcement learning. Whilst reinforcement learning and evolutionary algorithms are different methods they both look to do a similar thing, find a certain policy (behaviour) to maximise some reward (fitness). In combining learning with evolution more efficient and better performing solutions could potentially be found.

It is clear from all the studies looked at in this review 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 its variants in combination with local competition was a key focus of this review due to it offering a good overall solution. It may not always be able to outperform the fitness values of certain objective-based searches or even MAP-Elites but still offers a good result and is able to provide a wide diversity of solutions. This technique 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.

7. 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] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms. ACM Computing Surveys 45, 3 (2013), 1–33. DOI:https://doi.org/10.1145/2480741.2480752

[4] Giuseppe Cuccu and Faustino Gomez. 2011. When Novelty Is Not Enough. Applications of Evolutionary Computation 6624, (2011), 234–243. DOI:https://doi.org/10.1007/978-3-642-20525-5_

[5] Matthew Dale, Julian F. Miller, Susan Stepney, and Martin A. Trefzer. 2019. A substrate-independent framework to characterize reservoir computers. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 475, 2226 (2019), 20180723. DOI:https://doi.org/10.1098/rspa.2018.0723

[6] Yamille del Valle, Ganesh K. Venayagamoorthy, Salman Mohagheghi, Jean-Carlos Hernandez, and Ronald G. Harley. 2008. Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. IEEE Transactions on Evolutionary Computation. 12, 2 (2008), 171-195. DOI:https://doi.org/10.1109/TEVC.2007.89668624

[7] Ahmad Din, Meh Jabeen, Kashif Zia, Abbas Khalid, and Dinesh Kumar Saini. 2018. Behavior-based swarm robotic search and rescue using fuzzy controller. Computers & Electrical Engineering 70, (2018), 53–65. DOI:https://doi.org/10.1016/j.compeleceng.2018.06.003

[8] 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

[9] Hancong Duan, Chao Chen, Geyong Min, and Yu Wu. 2017. Energy-aware scheduling of virtual machines in

heterogeneous cloud computing systems. Future Generation Computer Systems 74, (2017), 142–150. DOI:https://doi.org/10.1016/j.future.2016.02.016

[10] Abdul Qadir Faridi, Sanjeev Sharma, Anupam Shukla, Ritu Tiwari, and Joydip Dhar. 2018. Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment. Intelligent Service Robotics 11, 2 (2018), 171–186. DOI:https://doi.org/10.1007/s11370-017-0244-7

[11] 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

[12] Jorge Gomes, Paulo Urbano, and Anders Lyhne Christensen. 2012. Progressive Minimal Criteria Novelty Search. Lecture Notes in Computer Science 7637, (2012), 281–290. DOI:https://doi.org/10.1007/978-3-642-34654-5_29

[13] 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

[14] 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

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

[16] Eisuke Kita. 2011. Evolutionary Algorithms. IntechOpen.

[17] 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

[18] Joel Lehman and Kenneth O. Stanley. 2010. Revising the evolutionary computation abstraction: minimal criteria novelty search. In Proceedings of the 12th annual conference on Genetic and evolutionary computation (GECCO '10). Association for Computing Machinery, New York, NY, USA, 103–110. DOI:https://doi.org/10.1145/1830483.1830503

[19] 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

[20] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. CoRR abs/1504.04909, (2015). DOI:https://doi.org/10.48550/ARXIV.1504.04909

[21] 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

[22] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. 2020. Swarm Robotic Behaviors and Current Applications. Frontiers in Robotics and AI 7, (2020), 36. DOI:https://doi.org/10.3389/frobt.2020.00036

[23] 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

[24] Giandomenico Spezzano. 2019. Swarm Robotics. MDPI - Multidisciplinary Digital Publishing Institute.

[25] Youchuan Wan, Mingwei Wang, Zhiwei Ye, and Xudong Lai. 2016. A feature selection method based on modified binary coded ant colony optimization algorithm. Applied Soft Computing 49, (2016), 248–258. DOI:https://doi.org/10.1016/j.asoc.2016.08.011

[26] 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

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

[28] 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

[29] 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

[30] 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

[31] 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.

[32] 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

[33] 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

[34] J.-B. Mouret and S. Doncieux. 2012. Encouraging Behavioral Diversity in Evolutionary Robotics: An Empirical Study. Evolutionary Computation 20, 1 (2012), 91–133. DOI:https://doi.org/10.1162/evco_a_00048

[35] Richard S Sutton and Andrew G Barto. 1998. Reinforcement Learning: An Introduction (1st ed.). Cambridge, Mass. : MIT Press.