UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2022

Title:

Hybridising Novelty Search with mEDEA algorithm for collective foraging

Author:

Sihle Khanyiso Calana

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*	15
System Development and Implementation	*0*	*20*	10
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*	
Total marks		**80**	

Hybridising Novelty Search with mEDEA algorithm for collective foraging

Sihle K. Calana
University of Cape Town
CLNSIH001@myuct.ac.za

ABSTRACT

The aim of this research to be done in this paper will be to find ways to improve how evolutionary collective robotic systems adapt to changes in their environment, whilst still solving problems without compromising on the robotic swarms behavioural diversity as this has proven to be difficult in past research. This issue will be approached by hybridising the mEDEA and Novelty Search evolutionary algorithms. This should addressed the trade-off between objective fitness and environment-driven fitness within a robotic swarm. The Novelty Search-mEDEA hybrid algorithms performance will be assessed on an evolutionary robotic swarm (with robotic controllers running the hybridised algorithm) behaviour when completing collective foraging tasks in a changing environment. This work would provide a basis for future studies and possibly physical implementation of collective construction using evolutionary swam robotics.

1 INTRODUCTION

Evolutionary robotics is an approach to robot development in the absence of human interference in such a way that machines are capable of learning in a decentralised manner [16] [8]. This means that there is no command centre to control all the robots in a swarm. Instead the behaviour of each robot is influenced by how they respond with their environment and each other [11]. This aids in designing flexible, robust and scalable collective behaviour patterns to coordinate a large number of robots to solve complex tasks [15]. This is useful for automatically generating artificial brains and simulated worlds that imitate real life, allowing for unconstrained exploration and testing of scientific hypotheses of biological processes [7] .

The way in which a collective group of robots work together to complete a task emulates how a colony of ants cooperate when foraging or a how a flock of birds collaborate when navigating and exploring the skies or how a school of fish communicate with one another. [19][14]

Now we can better understand Swarm Intelligence. Swarm Intelligence is a set of computing algorithms comprised of several multi-agent systems that mimic swarms in nature such as a swarm of bees [18]. These robots may be embodied or simulated. Experiments involving both are briefly discussed below.

This could be extremely useful in experimentation with regards to answering questions relating to evolution in the natural world, not just in artificial life [7]. This also greatly assists in completing hazardous tasks that may be harmful to human beings like underwater construction, humanitarian de-mining and space exploration (Cluster II mission) or difficult to reach such as search and rescue (Guardians project) to name a few [15].

The Cluster II space mission makes use of four unmanned aerial vehicles (UAVs) with aggregation and coordinated motion behaviour capabilities. This means they are able to move in formation and congregate in outer space. The 104 robot Perdix UAVs are another example of swarm robotics, used in the military for surveillance. These robots are capable of aerial navigation and decision making.[14]

Evolutionary computing is a section of computer science that uses biologically inspired natural evolution techniques to solve problems. When referencing biological evolution we are referring to an environment where the fittest individuals in the population are the ones that are most successful at surviving and passing their genes on to their offspring through reproduction. In the context of evolutionary algorithms, this would be called crossover [6].

In the evolutionary computing context, the fitness of robotic agents (individuals) would be determined by the fitness function which measures the progress towards the objective in a search space. This includes objectives such as to multiply and survive in an environment or to maximise exploration of the current environment. The evolution and adaptation of a robotic controller is dependent on evolutionary algorithms which mirrors natural selection [12] [17].

There are different controller models that can be used by a robotic swarm, for example, a simple reflexive agent model which exhibits purely reactive behaviour, or the model based reflexive agent, which similarly to the simple reflexive agent model, its behaviour is determined by reacting to its environment, however, unlike the simple reflexive model, it takes into account the history of its actions before making a decision.[21]

Living organisms are comprised of physical and behavioural features that determine how they interact with others and their surroundings. These attributes, called phenotypes, are determined by genes and the alleles attributed to those genes. Much like living organisms, robotic agents have invisible code (like genotypes) that influence their observable actions (similar to phenotypes). The fittest robots are more likely to have their genotypes passed on during crossover, which is the combination of features from two individuals in the offspring [6].

How well these agents adapt to their environment and/or task is greatly influenced by the basis for choosing a mate to reproduce with. This may be based on the degree of success with regards to the fitness function or simply exhibiting unique behaviours in an environment.

The issue here is that sometimes objective based selection does not always align with environment driven selections. For example, agents tasked with patrolling an environment requires them to be spread out, which may counter environment driven selection pressure to be close enough to mate [20]. Automatically adapting functional diversity (and collective behaviour) on experimental evolutionary robotics platforms has proven to be a challenge. There are a number of evolutionary algorithms to maximise behavioural diversity such as Map-Elites, Novelty Search and to name a few.

1.1 Prior Work

Briefly mention the papers from the literature review and related research done on evolutionary algorithms (and hybridising them to get new ones) as well as on swarm robotics

To gather more information on the field several research papers had been reviewed first. The experiments done in these papers were used as a jumping off point for our own research, in hopes of further developing the adaptive collective robotics field.

The paper by Bredeche and company, on environment based adaption of autonomous robots, served as the inspiration for the mEDEA algorithm we used in our experiments. Every agent within the robotic population had the algorithm though with varying genomes (collection of genes) [4].

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
```

The algorithm involved no selection bias, as a result of as the selection operator would randomly choose a mate from the list of imported genomes. Of course the more common the genome is the likelier it would spread. The variation operator would then mutate the selected genome before the replacement operator replaces the current genome with the mutated one. All of this is perfectly illustrated below.

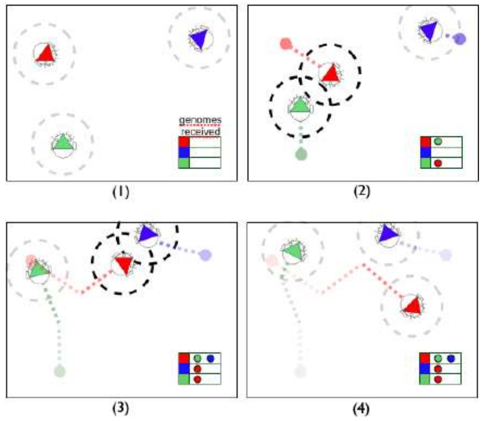


Figure 1: In (1) at the start of the generation the genomeList is empty. Each robot only contains their active genome which determines its behaviour. As they traverse the environment and come in contact with one another they exchange genomes when they're in close proximity. The red agent genomeList includes the green and blue agents genome from (2) and (3) respectively, whereas the other two agents both contain the red genome from their import list by the end of the generation in (4). The red genome spread more so has a greater likelihood of survival (randomly selected for mutation and passing onto the next generation). Green and blue only have a 50% chance probability of being mutated and carried on or being deleted

The mEDEA algorithm was the researchers attempt at addressing the problem, of conflicting fitness function motivations, mention briefly mentioned earlier. Specifically, the problem of finding balance between searching for other robots to mate with (described as the "intrinsic motivation") whilst maintaining survival efficiency (described as the “extrinsic motivation”). Simulated experiments were in two environments each designed to optimise one of the fitness function motivations. The first environment was the “free-ride” set up containing one hundred robotic agents with no resources. The agents were not distracted by energy resources, allowing them to focus on mating. This was run for the first 75 generations

The second environment was the “energy” setup which included food items that provide the agents energy, thus prolonging its lifetime. This would give it more time to spread its genomes, however time spent foraging meant time taken from mating. This environment was designed to measure how well the algorithm handled the trade-off between mating and foraging. This was done for the next 75 generations[4].

The experiment saw an initial drop in performance indicators (number of active agents, food items stored and mates), however this was followed by a prompt recovery due to the robots adapting and evolving. In both set-ups genomes from later generations were more probable to survive.

The experiment did not encourage collaboration between robots and they are either mating of gathering resources to sustain themselves. This is something we have accounted for when developing our experiments, discussed in the next section.

Another paper also involving Nicholas Bredeche, involved an experiment using 20 physical e-puck robots in an arena, each running the mEDEA algorithm. This was aimed to closed the reality gap between the simulation and the real world.

These robots were places in an arena with an obstacle that acted as the sun. Each robot know the suns location. The sun proved no selection advantage. Over-time the experimenter would arbitrarily move the suns location. The goal of this experiment was to study the how a population evolves towards a behavioural strategy within an environment with one singularity. After sometime more than half the robots in the environment were striving to stay in the ”sun”[5]. This was behaviour that was developed over time that was not there initially. The experimenter had compared this result to some prior experiment they had done using 10 robots and noted that experiments with larger populations tend to be more stable and less prone to extinction.

Number of hardware issues, that are often over looked when running simulations. The proximity sensors were unreliable and the genome exchange was often inconsistent. Nonetheless, the experiment still demonstrated that smaller robot population where less likely to converge towards a behavioural consensus than larger population samples. This proved mEDEA is robust to the reality gap despite the hardware challenges [5].

The magnitude of the reality gap between simulation could be further minimised through the use of more robust controllers. This could be done by maximising mutual information between simulated robot sensors and motors, thus promoting reactivity [22].

Novelty Search awards robotic agents in a swarm for performing a unusual, unique or out of the ordinary behaviour instead of behaviour based on an objective fitness functions [2]. This is especially useful when solving problems that have deceptively clear solutions. For instance, the goal of the Chinese finger trap is to free your fingers. It is ones natural instinct to want to pull your fingers away from each other (the objective), however this can further entrap your fingers. Only when attempting something uncommon and original (like pushing your fingers closer and further apart) is one able to actually get closer to the true objective. When it comes to working with robotic agents in an swarm intelligence, novelty search helps to provide an additional method of evaluating evolutionary algorithm solutions besides the objective fitness function. It also improves functional diversity in a robotic swarm by encouraging more exploration of the environment [3].

In one paper an experiment was conducted using maze navigation where agents were trained to reach a destination whilst avoiding obstacles. The maze navigation experiment involved using robots to make their way out of the maze to the final destination. Each robot had a built in radar to determine the distance to the closest obstacles, which acted as a compass by guiding the agent towards the goal [1]. There were 2 mazes, a medium and hard map.

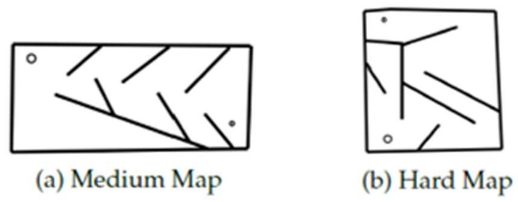
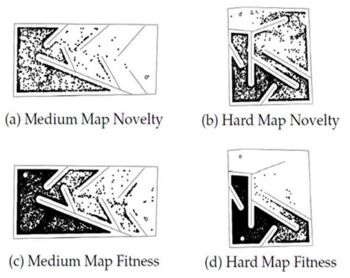


Figure 2: In each of the figures above, the smaller circle represents the agent and the larger one represents it’s destination. Using a straight forward, direct approach to the destination often led the agent into corners and cul-de-sacs preventing it from leaving the maze and reaching the destination [1].

The experiment was either run for 250 000 evaluations or once a solution was found (whichever came first). Scatter points were also made to track the traversal of a robotic agent as it searches for the exit to the maze.


(a) Medium Map Novelty (b) Hard Map Novelty

(c) Medium Map Fitness (d) Hard Map Fitness

Figure 4: **Final Points Visited Over Typical Runs.** Each maze depicts a typical run, stopping at either 250,000 evaluations or when a solution is found. Each point represents the end location of a robot evaluated during the run. Novelty search is more evenly distributed because it is not deceived.

Figure 3

The objective based NEAT algorithm often lead robotic agents into corners in pursuit of the final destination. Novelty based NEAT algorithms lead the robot to compare decisions to previous ones and move in the direction that has not been attempted yet. This was helpful in avoiding getting trapped in corners. Consequently, NEAT with Novelty Search reached the end of the medium difficulty maze 3 times faster (with 18.3 thousand evaluations) than fitness oriented NEAT algorithm (56.3 thousand) and with two-fifths of complexity. In the case of the hard difficulty map, the fitness objective function found the exit to the maze in only three out of forty runs, even getting surpassed by NEAT with random selection which yielded four solutions out of 40 runs. On the other hand, NEAT with Novelty Search achieved 39 solutions for 40 runs. Novelty Search exhibits an even distribution of points throughout both mazes, whereas, objective based NEAT algorithm displays areas of density around the local optima – which is commonly a problem for objective based fitness functions. Sometimes seemingly moving further away from the goal at certain parts of the maze actually improved the chances of reaching the objective [1].

There were some potential drawbacks of Novelty Search as maintaining an archive of past behaviours which can become computationally expensive, defining novel behaviour can be challenging especially in new and evolving environments, and lastly, performs rather poorly in an unrestrained environment as observed by a version of the maze experiment that was run without borders where only 5/100 evaluations found a solution.

There were several other papers in the collective evolutionary robotics field involving a range of experiments and tests, such as hybridizing novelty search with five variants of the HyperNEAT Neuro-Evolution for Transfer Learning (learning a complex task by learning several smaller ones)[10] and a few on the subject of embodied evolution such as the self-assembly experiments using physical kilobots to create formations[24].

1.2 RESEARCH OBJECTIVES

We look to expand upon the research we have reviewed and optimise our research methods based on the inefficiencies in the reviewed research. We want robots to be effectively applied to real life scenarios in a such a way that they are able to adapt to evolving environments whilst still being able to efficiently complete the task at hand. We aim to automate the adaptation of robotic controllers used for generating co-operative behaviour to solve problems in evolving environments. We aspire to achieve this by hybridising evolutionary algorithms to optimised behavioural diversity. Our goal is for the robots to adapt to changes in the environment, whilst still completing the objective

A simulation of e-puck robots will be tasked with foraging resources (of different weights and sizes) and storing them in a gathering zone. The robots will use a hybridization of the minimal Environment-driven Distribution Evolutionary Adaptation (mEDEA) algorithm and Novelty Search diversity algorithm. The methodology outlines the design and implementation of our experiments as well as the resources utilized in them. The results are then graphed out and compared to non-hybridized cases before conclusions are drawn on the level of success of our research.

2 METHODOLOGY

The robot simulator, called roborobo was used to run our experiments. The simulator was extracted from N. Bredeche GitHub repository. A lot of the physics for how resources and robots interacted with the environment was built in and mostly coded in C++ and uses the SDL2 graphics library[25]. The code concerning the experiments was coded in python. The experiment was run in a 600 pixel by 600 pixel frame and each robots was equipped with 8 sensors, 24 pixels long. Robots were tasked with navigating the environment, looking for resources and taking the resources to the gathering zone, which was located at the bottom right of the arena of the arena. Even though each robot ran the mEDEA algorithm their genomes were randomly initialized so the robots each had their own individual behaviour. As a result of there being a large set of autonomous components involved and the distributed processing among them. Man, the media algorithm seemed the best, most suitable. Evolutionary algorithm to use. [5]

In very basic version of the media algorithm was provided to us, although it needed some adjustment to make it suit our specific objectives and the task at hand. This adjustments included adding a variation method to mutate genes, followed creating a selection operator since the existing one was simply a random decision. This completed the mEDEA implementation. After that we created the functions needed to support the use of Novelty Search. Then we added pick up and drop heuristics to the robot controllers so that the robot responds as expected to reaching an object or the storage drop zone. Finally we created a means for the simulation to print out the results to a text file for anyone running to view it.

While the robots were not carrying resources to the gathering zone, they were free to move as their wanted based of course based on their genomes. There was a limited level control Over how the robots behaved when they were not carrying a resource. This was to allow avoid interfering in the adaptation of the robot controllers. Of course there was a still Obstacle avoidance when it comes to getting stuck in walls and Obstructing other robots carrying resources, however there was minimal hand coding. The meeting and exchange of genomes of the robots was based on proximity. If a robot detected another robot on its sensor, they would mutually exchange genomes. The mutual exchange was to ensure that the robot's controller was as robust as possible [22]. Each robot had a lifetime of 600 iterations. The robots were coded with heuristics on what to do if they encountered a resource, which was to take it to the gathering and another heuristic if the resource was too heavy, which was to wait for another robot to help them out, but only for a certain period of time before they abandon the resource to look for other resources. This was to avoid a robot wasting its lifetime trying to move a heavy resource without help ever arriving. This was also inspired by one of the research papers I had read earlier [26].

The resources that had to be gathered had different sizes and weights. A type A resource only needed one robot to push it to the gathering zone, whereas the type B resource needed two robotic agents, and a type C required three robots to push it to the gathering zone. This was done to enforce collaboration between the robot so they would be required to work together to achieve the primary objective. This was done with our goal of adapting collective behaviour goal in mind. The type A resources where six pixels in radius and were coloured green. The typing resources were ten pixels in radius and were coloured blue. The Type C resources were twelve pixels in radius and they were coloured red.

Since the primary objective was to gather resources, we based objective fitness function on the number and the type of resource that we gathered. A type a resource was worth one waiting, a type B resource was with two, and a Type C resource was with three. Steve, Objective fitness of a robot was determined by the sum of the weightings of the resources they had gathered in the lifetime. The behavioural fitness was determined based on two metrics. One was using the novelty search, with regards to the Euclidean distance and the other was the number of genomes that they had acquired during their lifetime. The behavioural fitness Function was determined by multiplying the two metrics. To clarify the number of received genomes a robot head, the number of resources an agent stored and the novelty with regards to the average sum of the Euclidean distance from a robotic agents neighbours were all metrics that contributed towards selection bias. The mEDEA algorithm was run with objective fitness function In an environment. Then media algorithm was run on the same environment again, except with the behavioural fitness function. The simulation was then run a third time with the random selection this time.

Three different versions of the media algorithm were each written in their own separate Python module. They imported a common module that was also created separate from them, which was which contained the Resource class and the food item class, as well as the main method. At the end of each generation, they would evaluate the respective of fitness function and would then. Make a selection based off of that. Based on whichever genome was chosen as a slight Variation function is applied to the genome to mutate the genes Thus creating a new robot behaviour. Each robot had its own colour as well, which was also varied to express a combination of both parents colours. This was done using RGB colour components.

Two separate environments we used to run all three of the different versions of this simulation. The first environment involved 200 robots and 100 resources. A large number of robots were chosen to ensure the number of deactivated robots were kept to a minimum. The resources were various types of course, and we chose a large number of resources to be in the environment to ensure as many robots as possible were at least encountering a resource. This was also done to in an attempt to optimise the objective fitness value of the robot. The robots were ran in this environment for 20 generations and ran 15 times.

The second environment was slightly different. There were a lot less robots in the environment as it used 100 robots and only 50 Through all the generations. Even then, the resources were not released to the environment all at once. The only 15 resources were released to the environment for the first 5 generations and then, in the following five generations, another 35 resources were released. This was to give time to the robots to adapt and navigate their environment before gathering resources. The time was measured for how long it took us to gather resources in the first five generations, and it was compared to the time taken to gather the same number of resources in the following five generations - after the robots had adapted a little to the environment. The waiting period before adding more resources was also inspired by one of the earlier experiments for the mEDEA algorithm mentioned earlier in the paper [4]. This was also done to measure how well the robots adapt to changes in the environment.

EXPERIMENT TABLE	
Parameter	**Value**
Arena Size	600x600px
lifetime	600 iterations
Resource Size (simple case)	6px
Resource Size (intermediate case)	10px
Resource Size (complex case)	15px
Number of Robot Sensors	8
Robot Sensors Range	24px
Controller Type	Multi-Layer Perceptron
Number of Hidden Layers	1
Neurons per Hidden Layer	3
K-nearest Neighbours (for Novelty Search)	10
Number of Robots for Experiment 1	200
Number of Iterations for Experiment 1	12 000
Number of Generations for experiment 1	20
Number of Resources for experiment 1	100
Number of Robots for Experiment 2	100
Number of Iterations for Experiment 2	6 000
Number of Generations for experiment 2	10
Number of resources for experiment 2	50

3 RESULTS AND DISCUSSION

After the reviewing the results from the outputted textfiles for each of the fifteen experiments and taking an overall average we noticed that by the end of the twentieth generation for the simulation ran in the first environment, we noticed that the mEDEA algorithm with random selection had the most active robots out of the initial 200 agents that were added to the environment. This is illustrated by the graph below.

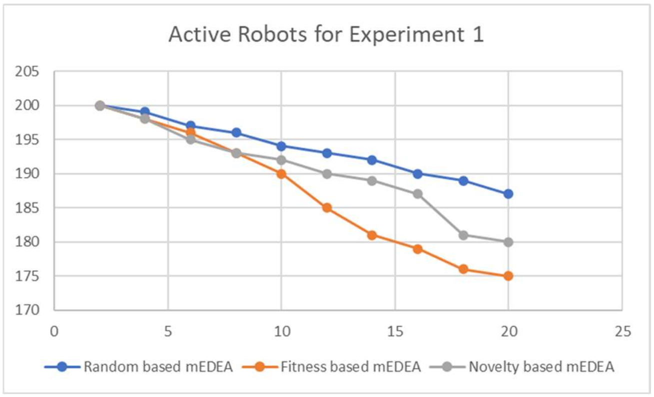


Figure 4: The number of active robots over 20 generations in Experiment 1

This could be as a result of robots running on the random based mEDEA algorithm constantly exchanging genomes with other robots also circling the same spot of in the environment. This may also have been due to Fitness and Novelty based mEDEA algorithms making the robot controllers better adapt to completing the task. This is supported by the fact that on average the other 2 algorithms stored more items.

The graph below shows the objects stored per generation over the 15 runs.

Generation	Random based mEDEA	Fitness based mEDEA	Novelty based mEDEA
2	26	23	24
4	27	24	26
6	30	26	30
8	33	28	34
10	36	33	40
12	41	40	57
14	46	48	63
16	51	58	66
18	57	65	72
20	57	65	75

The table shows that the fitness and the novelty search based algorithms performed very similarly. Even though the hybridized Novelty Search algorithm person slightly better in terms of completing the objective. This may be a result of the Novelty robot controllers having evolving to explore their environment and thus encountering more robots and resources. This would also justify why the fitness based Algorithm also has less active robots. This is a step in right direction toward our goal for the Novelty-based mEDEA algorithm to handle the offset between getting swarm robotic agents to adapt to their environment and completing the objective. Though this is the result we expected, the magnitude of the improvement was rather minimal. This may have been due to the robot controller not maintaining a larger enough archive of its past actions when it came to determining novelty. This would be of great interest if it could be improved and explored further.

The results for the second experiments are displayed in the table and illustrated in the chart below. In the table we see the results for objects stored for the first 5 generations (with 15 resources present) and then for the following 5 generations (with an additional 35 resources added to the environment. The time below is show in seconds.

Resource storage for experiment 2

Algorithm (-based mEDEA)	Number of Generations	Number of Objects Stored	Storage Value	Time taken to store 10 resources
Random	5	8	11	118
	10	20	29	40
Fitness	5	10	18	122
	10	24	40	31
Novelty	5	11	13	121
	10	35	51	28

All three algorithms score roughly the same time (about 2 minutes) for storing resources during the first five generations. From what was observed when running the simulations, most of the robots wasted time during the early simulations. Often circling about the same spot. This is greatly improved upon in subsequent generations, evident by the time taken (see table) to store 10 resources after the 5 generation. It can be seen in the chart that random selection performed similarly to the fitness based algorithm. This may be because the robots occasionally choose the optimum mates genome from time to time. This could also be attributed to resources spawning in favourable positions like near robots and away from walls.

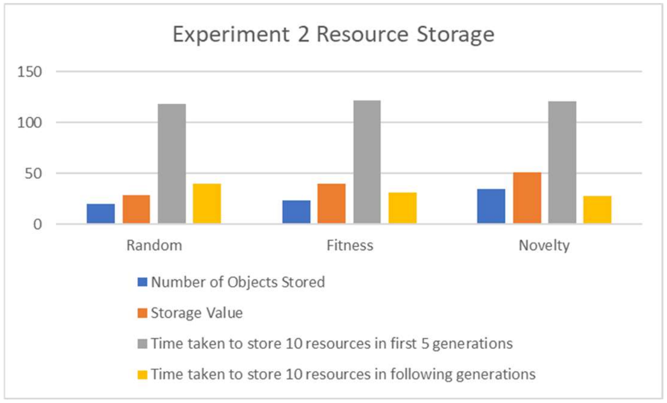


Figure 5: Compares the number, Value and Time of the resources stored in experiment 2

There was a significant jump in resources stored for the mEDEA algorithm with Novelty Search between the 5th and the 10th (final) generation. This is in support with our aim to determine how well this hybridized algorithm can adapt robotic controllers to changes in the environment whilst still optimising objective completion. Looking at the table we notice that the novelty-based mEDEA algorithm has the highest storage value. We could conclude that this is due to the robots in the swarm being more collaborative, thus storing heavier resources – which have higher value. This is reinforced by the fact that robotic controllers running on the hybridised algorithm stored more resources.

4 Conclusions

Overall will this help with physical applications in the real world (not just in simulations)

Overview of the success or failure of the experiment with regards to our expectations
How effective hybridising evolutionary algorithms are and its other potential uses such working with artificial brains.

The results align with the objectives of our research stated earlier. The hybridised algorithm - mEDEA algorithm with Novelty Search – appears to handle to conflicts between environment and objective driven selection by efficiently adapting to the environment (exploring and spreading its genome) without compromising on completing the objective (gathering resources) as seen in the first experiment environment. The Novelty-based mEDEA algorithm also indicated improved collaboration between robots as they brought in more resources and this had a higher overall storage value. It also seemed to optimise adaption to the changes in the environment as deduced from the second experiment environment. The effectiveness of the hybridisation should be tested further as only 25 tests were runs in total due to time constraints. It would be of interest to the field of evolutionary swarm collaborative problem solving to see how controllers using the hybridisation operate in collective formation tasks. This combined with foraging may be of great use for building construction in the future. Of more tests would need to be done on physical robot.

ACKNOWLEDGEMENTS

I would like to acknowledge professor Geoff Nitchke with providing we with the resources needed to acquire the robot simulator as well as the advice he would provide throughout the project, as well as the understanding towards the difficulties I faced whilst completing the project. I would also like to acknowledge my teammates Bailey Green and Kevaalin Rapeti for assisting me when they could while setting up the simulator.

REFERENCE

[1] Lehman, J. and Stanley, K., 2011. Abandoning Objectives: Evolution Through the Search for Novelty Alone. Evolutionary Computation, 19(2), pp.189-223.

[2] Gomes, J., Urbano, P. and Christensen, A., 2013. Evolution of swarm robotics systems with novelty search. Swarm Intelligence, 7(2-3), pp.115-144.

[3] Fister, I. et al., 2019. Novelty search for global optimization. Applied Mathematics and Computation, 347, pp.865-881.

[4] Bredeche, N., Montanier, J., Liu, W. and Winfield, A., 2012. Environment-driven distributed evolutionary adaptation in a population of autonomous robotic agents. Mathematical and Computer Modelling of Dynamical Systems, 18(1), pp.101-129.

[5] Bredeche, Nicolas & Montanier, Jean-Marc. (2012). Environment-driven Open-ended Evolution with a Population of Autonomous Robots.

[6] Eiben, A. and Smith, J., 2015. Evolutionary Computing: The Origins. Natural Computing Series, pp.13-24.

[7] Floreano, D., Husbands, P. and Nolfi, S., 2008. Evolutionary Robotics. Springer Handbook of Robotics, pp.1423-1451.

[8] Boumaza, A., 2022. Seeking Specialization Through Novelty in Distributed Online Collective Robotics. Applications of Evolutionary Computation, pp.635-650.

[9] Didi, S. and Nitschke, G., 2016. Hybridizing novelty search for transfer learning. 2016 IEEE Symposium Series on Computational Intelligence (SSCI).

[10] Haasdijk, E., Bredeche, N. and Eiben, A., 2014. Combining Environment-Driven Adaptation and Task-Driven Optimisation in Evolutionary Robotics. PLoS ONE, 9(6), p.e98466.

[11] Illuminating search spaces by mapping elites

[12] Schranz, M., Umlauft, M., Sende, M. and Elmenreich, W., 2020. Swarm Robotic Behaviors and Current Applications. Frontiers in Robotics and AI, 7.

[13] Swarm robotics: a review from the swarm engineering perspective

[14] Pini, G., Brutschy, A., Frison, M., Roli, A., Dorigo, M. and Birattari, M., 2011. Task partitioning in swarms of robots: an adaptive method for strategy selection. Swarm Intelligence, 5(3-4), pp.283-304.

[15] Evolution of a functionally diverse swarm via a novel decentralised quality-diversity algorithm

[16] Werfel, J., Petersen, K. and Nagpal, R., 2014. Designing Collective Behavior in a Termite-Inspired Robot Construction Team. *Science*, 343(6172), pp.754-758.

[17] Barca, J. and Sekercioglu, Y., 2012. Swarm robotics reviewed. Robotica, 31(3), pp.345-359.

[18] Doncieux, S., Bredeche, N., Mouret, J. and Eiben, A., 2015. Evolutionary Robotics: What, Why, and Where to. Frontiers in Robotics and AI, 2.

[19] Hamann, H., 2018. Short Introduction to Robotics. Swarm Robotics: A Formal Approach, pp.33-55.

[20] Lehman, J., Risi, S., D'Ambrosio, D. and O Stanley, K., 2013. Encouraging reactivity to create robust machines. *Adaptive Behavior*, 21(6), pp.484-500.

[21] Rubenstein, M., Cornejo, A. and Nagpal, R., 2014. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198), pp.795-799.

[22] N. Bredeche, J.-M. Montanier, B. Weel, and E. Haasdijk. Roborobo! a fast robot simulator for swarm and collective robotics. CoRR, abs/1304.2888, 2013.

[23] A.Furman, D. Nagar, G. Nitschke. "Automating Collective Robotic System Design"