Evolving Agent-based Models Using Complexification Approach

Wagner, M.; Cai, W.; Lees, M.H.; Aydt, H.

DOI
10.1016/j.procs.2014.05.028
Publication date
2014
Document Version
Final published version
Published in
Procedia Computer Science

Procedia Computer Science

Volume 29, 2014, Pages 310–321

ICCS 2014. 14th International Conference on Computational Science

Evolving Agent-based Models Using Complexification Approach

Michael Wagner[1], Wentong Cai[2], Michael H. Lees[3], and Heiko Aydt[1]

[1] TUM CREATE, 1 Create Way,
Singapore 138602
[2] School of Computer Engineering, Nanyang Technological University,
Singapore 639798
[3] Informatics Institute, University of Amsterdam,
Science Park 904, 1098 XH Amsterdam, the Netherlands

Abstract
This paper focuses on parameter search for multi-agent based models using evolutionary algorithms. Large numbers and variable dimensions of parameters require a search method which can efficiently handle a high dimensional search space. We are proposing the use of complexification as it emulates the natural way of evolution by starting with a small constrained search space and expanding it as the evolution progresses. We examined the effects of this method on an EA by evolving parameters for two multi-agent based models.

Keywords: Evolutionary Computing, Agent-based Simulation, Modelling, Artificial Life

1 Introduction

Agent-based models (ABMs) are among the most important tools for exploring emergent behavior, a phenomenon that describes the behavior of a system, which cannot be explained alone by the sum of its parts. Understanding and harnessing emergence is very important because it allows to create complex behavior, based on the interaction between relatively simple components. One way to discover and examine emergence in a computer simulation is to calibrate the model parameters accordingly. Underlying models often encompass a wealth of parameters, making the search of the sheer size of multi-dimensional space a problem. Due to interdependence and interaction between agents, slight changes in the model configuration can amount to very different simulation outcomes, indicating the high level of complexity. Even though Evolutionary Algorithms (EAs) are often designed and used to efficiently explore large parameter spaces, traversing those can still take a considerable amount of time. In this paper we propose the use of complexification to improve the performance of EAs used for parameter estimation of multi-agent based models. In an earlier work [12] we gave evidence that evolving parameters is directly influenced by the model's complexity. Therefore it is essential for EAs to be flexible

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.028

enough to handle complex models. Traditional EAs are forced to make assumptions about the problem. Properties like the length and structure of genomes have to be determined a priori and cannot be changed during the EA run. However, complex systems can have a variable number and structure of parameters. Rule-based ABMs face the same issues. Rules can consist of an arbitrary number of components like conditions and actions. This makes finding optimized solutions very difficult if those require a large number of components or complex rule sets.

Natural evolution is far more than just a cycle of recombining and mutating genes. In order to advance from basic single-cell lifeforms to complex organisms, the genotype of species has to be extended [3, 4]. Complexification is a substantial part of this process as it allows for an organism to increase its genome size and to become more versatile. By incrementally adding new genes organisms can, over time, adapt to their environment and develop further characteristics or skills. In nature this happens by gain-of-function mutation, a type of mutation that increases the genome size and confers new properties and eventually new functionality to the organism. The increase in size most commonly happens by a random duplication of parts of the genome.

By using complexification repeatedly the evolutionary process can cover a wider variation of potential traits and functions of the individuals subjected to it. We believe that complexification, as a form of incremental evolution, is able to improve the estimation of model parameters. We want to validate this by conducting experiments on two multi-agent based models where we test an Evolution Strategy against a redesigned version of itself which incorporates complexification. This allows for a direct comparison of the evolutionary techniques.

In the following parts of our work we first provide a short overview on the evolution of model parameters for ABM and complexification. Secondly, in section 3 the benchmark models for testing our complexifying EA are introduced, followed by a description of the EA and its mechanics itself in section 4. Finally in section 5 our experiments are described and concluded with a discussion.

2 Related Work

Evolution of model parameters in multi-agent based models has been pioneered in [6] and [13], among others. It is very useful as a tool to configure and explore the real life systems, which the models are based on. One of the major applications of it is to detect and explore emergent behavior that may arise. Emergent behavior is a phenomenon that requires and indicates the level of complexity of a natural or artificial system. In connection with ABMs it has recently been experimented with by [11]. In [12] the evolution of boid model parameters for discovering forms of emergence, in relationship to an objective fitness measure, was described. Furthermore it was discussed how the model complexity influences this process and what challenges it holds. We learned from this work that, while leaving the fitness measure unchanged, extending the model detail and thus increasing the number of possible configurations may have negative effects on the parameter evolution. The present work continues this train of thought and argues how the apparent difficulties can be approached. Complexification with regard to EAs has been applied almost exclusively in evolving artificial neural networks (ANNs) [9], benefiting from their easily modifiable graph-like structure. Other applications include the evolution of strategies for single 3D agents [10] successfully proved the superiority of incremental evolution by evolving the weights rather than structure of the ANN that is controlling the agents movement.

3 Agent-based Models for Experimentation

We chose two multi-agent based models to test our hypothesis: the boid model, introduced in [5] and its further development, the bee swarm model. The models we use are able to represent multiple species of agents to study cooperative and competitive behavior, which are important factors in the creation of emergence. The emphasis is put especially on their effect on the survival rate of the boids.

3.1 Boid Model

Our customized boid model, as described in [12], contains a simple predator-prey scenario. Additionally to the boids, which are now considered prey, it also involves predators and food sources for the boids, thus creating a simple food chain. The goal for which to optimize the model parameters, is to maximize the boid survival chance by having them graze the food sources and evade the predators as efficiently as possible. To add more strategic depth, the boids can be grouped in up to three different species where all members of one species have identical parameters. However, different species can have different parameters. This is a generalization of cooperation between multiple species as it occurs in nature among certain kinds of birds [7], among others. These species ignore each other by default when it comes to flocking behavior, but there are flags, called "alliances", which can be set to enable collective flocking. Boid species have two options: either cooperate or to try to survive on their own. For this model we define a fitness function $f_{survived}(\vec{x})$ in equation 1 as the number of boids which are still alive after the simulation terminates at simulation time t_{stop}. The argument $\vec{x}$ hereby denotes the parameter vector that will be used to initialize the simulation.

$$f_{survived}(\vec{x}) = |b \in aliveBoids_{t_{stop}}| \tag{1}$$

The challenge now is to find the following parameters, contained in vector $\vec{x}$, so that $f(\vec{x})$ is optimized:

1. The number of different boid species (in this case ranging from 1 to 3)
2. The five parameters regarding movement and flocking behavior: avoidance, convergence, cohesion, momentum and sensor range
3. The alliance flags between the boid species

3.2 Bee Swarm Model

We transformed the boids to a swarming model and added dependencies between them to make it more suitable for exploring complexification. According to its name, the model attempts to give a simplified presentation of how a bee swarm works. Firstly, the bees all originate from a hive, centered in the simulated world. In order to sustain their colony it is necessary to look for food, forage it and return it to the hive. The bees are facing two challenges here: food sources e.g., flowers, vary in the amount of food they provide and there are competitors present, eager to steal food from the hive.

Similar to the boid model, bees can be also divided into species. This emulates the natural division of labor into workers and drones. In contrast to nature, the groups can choose their

specialization by distributing points into three different skills: foraging, scouting and defending. Each skill may have between 0 and 10 points invested and has been artificially equipped with artificial caveats and benefits to create trade-offs and avoid the generation of perfect boids. Foraging skill increases carrying capacity but penalizes defense while the defending ability acts conversely. Scouting ability benefits sensory range and movement speed, but takes away from both, defense and foraging skills. These are the most important evolvable parameters in our model as they strongly influence the bees behavior. The number of bee groups as well as their share in the total bee population are two more evolvable parameters. Remaining parameters are the following five, similar to the predator-prey boid model: avoidance, convergence, cohesion, momentum and size of neighborhood.The challenge here is to find the following parameters, so that a given fitness function $f_{gathered}(\vec{x})$ in equation 2 is optimized:

1. The number of different bee groups (in this case ranging from 1 to 3)
2. The share of bees per group
3. The skill distribution per group
4. The five parameters regarding grouping behavior, as mentioned above

For our experiments we use a relative food gathering fitness $f_{gathered}(\vec{x})$, which expresses the relationship between gathered food and lost or spent food. Similarly to the boid model, $f(\vec{x})$ is to be maximized.

$$f_{gathered}(\vec{x}) = \frac{gatheredFood(hive)}{spentAndLostFood(hive)} \tag{2}$$

The relative food gathering fitness.

4 Complexification in Evolutionary Algorithms

The essence of complexification is to improve the potential fitness of individuals by gradually extending their genome size or structure to allow for more diversity and more possible directions to evolve, including specialization or improvement of existing features. Typically this will be achieved by increasing genotype size or changing the relationship between genotype and phenotype. The latter option means not to have a one-to-one relation between both, but rather a n-to-one relationship where genes can be reused and influence more than one phenotypical trait at the same time. The first option directly translates into adding more possible combinations of genes.

The minimum requirement for integrating complexification into an EA is a mechanism to extend the genome (see figure 1) and to ensure that crossover operations between EA individuals, which have been altered by complexification, are handled in a meaningful way [9]. The latter is especially important since a crossover may encounter individuals of different genome length, resulting from complexification. In that case there are different ways to execute the crossover, which include only matching genes common in both chromosomes or by a randomized method, among others. The difficulty here is to make sure that the resulting chromosome is valid. Another option would be to only allow crossover between chromosomes of the same length. In the following we briefly describe how complexification is supposed to work within the EA and give details about our implementation.

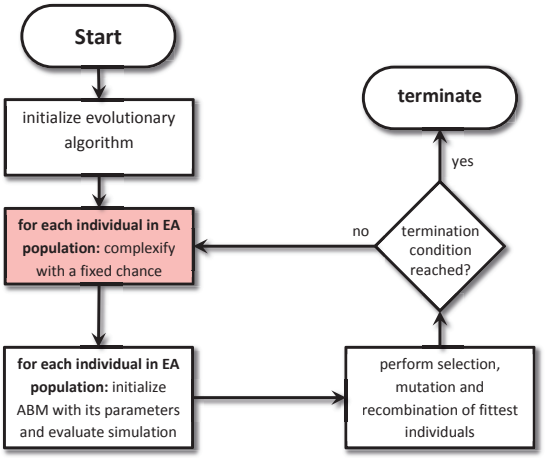

Figure 1: Scheme of evolutionary algorithm, entry point of complexification highlighted in red.

To implement and integrate complexification into our existing EA the following adjustments have to be made. Firstly, it has to be determined whether the chromosomes of the EA individuals are extendable and, if so, how the extension should be performed. As mentioned earlier, both scenarios use a real-valued representation which encodes the parameters for up to three groups of boid or bee agents, respectively to the scenario. The minimum requirement for a valid chromosome is an ability to encode at least one agent group. Based on this we can define the extension of a chromosome as the addition of genes that are necessary to encode another agent group. Secondly, we have to define how the extension of individuals is handled by the EA. Like in nature complexification will occur randomly, more precisely every individual has a chance per generation to be extended. This chance is equal for all individuals and set before the EA starts. In section 5 we will experiment with complexification chances of 1%, 5%, 10% and 30% per individual per generation to compare the effects of difference chances. Finally, the crossover has to be altered in order to be able to operate on a population with varying chromosome lengths. In this work we decide to modify the selection to allow an individual to only be crossed with another individual that has a genome of equivalent length. This way the difficulty of how to interbreed chromosomes of unequal length can be ignored. Interbreeding would necessitate a mechanism to decide how to treat genes of one parent without a counterpart in the other parent, if both are of different chromosome length.

As in [12] for the boid model, we again use the Java-based simulation framework MASON [1] to create the bee simulation. Likewise, the EA framework ECJ [2] is used again for evolving the model parameters. A disadvantage there is that the framework does not actively support a variable length of real-valued vector individuals. To compensate for this shortcoming, we supply all EA individuals with a full set of genomes, capable of encoding the maximum number of boid or bee groups. Depending on the complexity level, which is contained in one of the alleles, only the necessary parts of the genotype will actually be used and evaluated, as seen in figure 2.

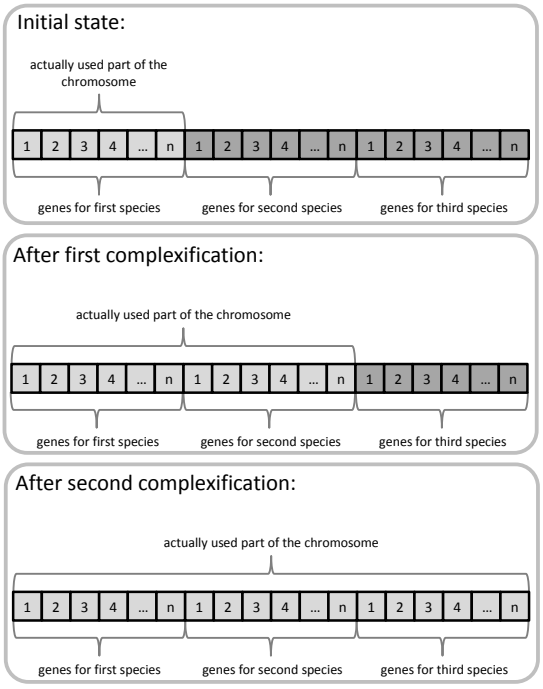

Figure 2: Complexification of an EA individual (chromosome string) in our boid and bee experiments.

5 Experiments and Results

In the following we will discuss the two experiments that had been conducted in order to validate the hypothesis. For each experiment a Monte-Carlo search was also executed, by sampling randomly generated solutions. This allows us to make assumptions about the distribution and fitness of the solutions in the search space. The first experiment concerns a model, that has already been explored by EA in [12], and will be explored now again with the aim to show how complexification can improve the results. Meanwhile the second experiment is supposed to illustrate the efficiency of self-adapting complexification over conducting multiple separate EA runs.

5.1 Boid Model

The survival scenario is, to a great extent, identical to the one described in [12]. There the EA was unable to find solutions for the two- and three-species configuration that were of equal quality as the ones found for the one-species configuration, even though the solutions of the latter are contained within the search spaces of all three configurations. In short: if every species in a multi-species solution has the same parameters, they will act as one species. Therefore every parameter search for a multi-species solution has the potential to reach a fitness of the same or better quality as a one-species solution. To summarize, the objective here is to show the effectiveness of complexification, as it is supposed to be able to explore the search space more efectively and produce multi-species solutions with at least the same fitness as single species

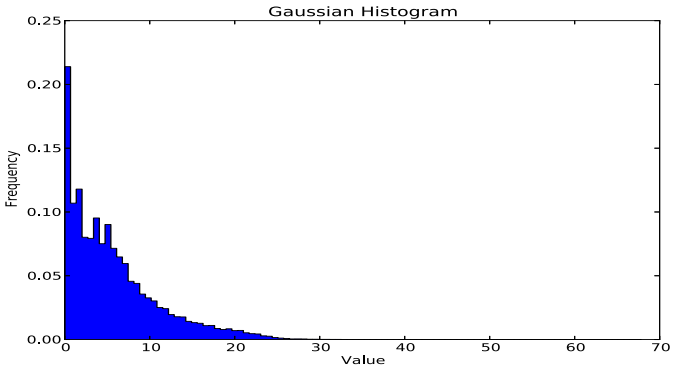

Figure 3: Monte Carlo Search on the survival scenario

solutions.

Monte-Carlo Search Since there are 70 boids in the scenario, the best fitness achievable is 70, given all of them survive. Our random search, as depicted in Figure 3, with a sample size of 250000 found solutions with fitness results of up to 67. This indicates that it is possible to find near perfect solutions. However, more than 90% of all samples do not exceed a fitness of 20. Therefore the major part of the solution space contains very weak solutions. This also becomes evident by the very low average fitness, as seen in table 1. Like the EA in [12], the best individual is a one-species solution. But on average the two-species solutions come out the strongest with three-species solutions as the second best.

complexity level	average fitness	highest fitness
1	1.33	67.8
2	3.03	40.48
3	1.44	28.6

Table 1: Survival Scenario: Average and best fitness results of random search.

Evolutionary Algorithms In our experiments we are using four different complexification probabilities. These are indicating the chance for an individual to increase its complexity every generation. The chances we used are: 1%, 5%, 10% and 30%. Higher values accelerate the complexification process and push the EA population faster towards higher complexity. Lower values on the other hand allow for more thorough exploration of low complexity levels by exerting less pressure on increasing the complexity. Both ways have advantages and disadvantages. So, it is important to find a middle ground between too high and too low of a complexification chance, which can maximize the quality of their solutions. If we compare random search and EA runs we can observe the weakness of the default EA compared to the version with complexification, as seen in Tables 2 and 3.

The comparison between the two configurations shows that complexification is superior to the default EA in both, average and maximum fitness values. One significant shortcoming of the EA described in [12] was the incapability to find high quality solutions within the space

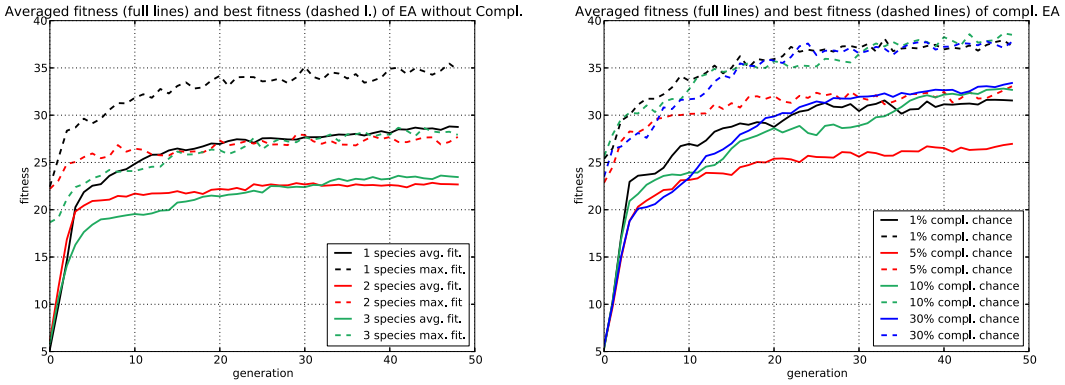

Figure 4: Left: survival scenario without complexification, right: with complexification dotted lines are standard deviation

fixed complexity level	highest fitness	avg. fitness
1	39.4	24.2
2	29.0	20.8
3	31.0	20.7

Table 2: Survival Scenario: Average and best fitness results of the EA without complexification

of multi-species individuals, even though they existed. With added complexification the EA is now able to find multi-species individuals that are qualitatively comparable to the one-species individuals. Another aspect is the population dynamic, in figure 5 we can see how a high chance complexification changes the composition of the population from entirely one-species individuals gradually to a majority of three-species individuals in the end.

5.2 Bee Swarm Model

The second series of experiments was conducted with the bee swarm model. In contrast to the boid model this one has been designed to require multiple bee species for achieving high quality solutions. What this means is that solutions with two and three species enable the bees to have more than just one forager or defender species and thus provide better strategies for the bees to fulfill their tasks as stated in section 3.2. As a result solutions with two or three bee species should be preferred by the EA. The objective here is to show that complexification can deliver a solution, with the same quality and in shorter time as an EA without complexification that is only searching on one specific complexity level at a time. At the beginning of the simulation 25 bees are placed on the two-dimensional field. The competitors spawn randomly, with a chance of 1% for one competitor per simulation tick, for a total duration of 3000 simulation steps. The fitter the bee skill sets are, the more efficiently they can keep their competitor numbers small and away from their hive, where their food reserves are stored.

Monte-Carlo Search Similar to the boid model we executed a random search to get an idea of the shape and structure of the fitness landscape. The result of 200000 samples indicates that

complexification chance	highest fitness	complexity	avg. fitness
1 %	72.0	1	35.0
5 %	71.0	1	28.1
10 %	70.4	2	23.3
30 %	69.2	3	27.2

Table 3: Survival Scenario: Average and best fitness results of the complexification runs

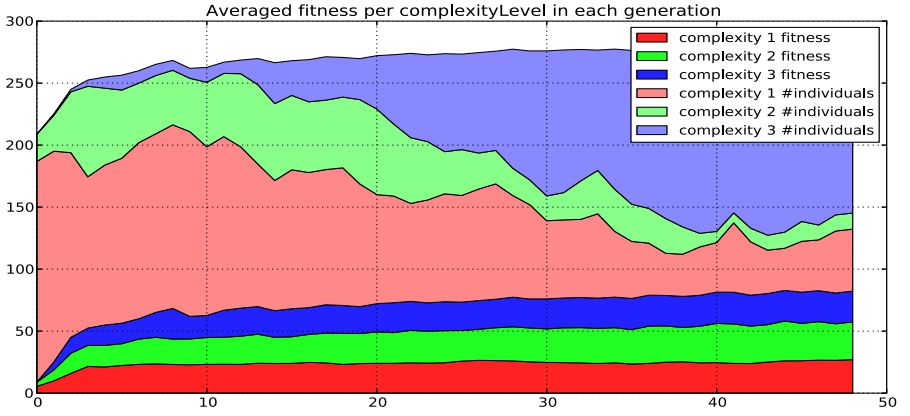

Figure 5: Survival Scenario:
light colors - relationship between number of individuals with respective complexity level
dark colors - relationship between fitness of individuals with respective complexity level

the fitness landscape for the most part consists of very low-fitness individuals. The parameter configurations of the samples suggest that solutions with higher fitness are mostly symmetrical, e.g., the bee species have similar parameters while the chromosome contains them in a different order. As a conclusion this indicates that there are more than one global optima. The random search shows how difficult it is to find individuals with above average fitness. Even though the best individual found has a fitness of 135856 the average fitness of all samples is vanishingly small, as seen in Table 4. This is a significant discrepancy, caused by the huge amount of samples with a low fitness as seen in Figure 6. Note that the x-axis starts with the fitness 2, because the number of individuals with fitness 0 and 1 are too large to be depicted on scale. This is even more emphasized by the very low frequency of fitter individuals found, as seen in those figures.

complexity level	average fitness	highest fitness
1	4.01	8762
2	64.27	135856
3	26.81	125523

Table 4: Gathering Food Scenario: Average and best fitness results of random search.

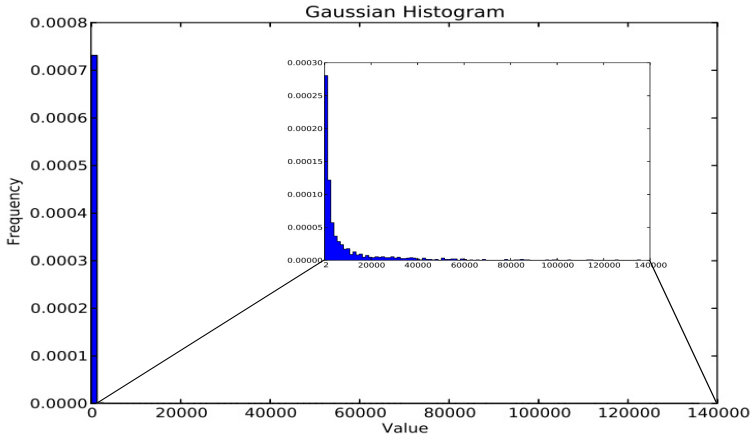

Figure 6: Random Search on the gathering food scenario. The inner diagram starts at x=2 and has a different scaling to show the huge difference in numbers between individuals with fitness of 0 or 1 and individuals with higher fitness.

Experiments For the following experiments the EA used a population of 50 individuals and evolved them over a span of 100 generations. The simulation to evaluate an individual's fitness is repeated as often as necessary for the standard error of the samples to fall below a predefined value. Each EA run is repeated 5 times and the results show the average over the repetitions.

Similar to the survival experiment, we again use EAs with four different complexification probabilities. Here we can observe interesting behavior: even though the average fitness achieved with the complexification is lower than that with the EA without complexification (see figure 7), the highest fitness-es is on an equal level (see tables 5 and 6). The advantage of complexification here is that its search covers the space of all complexity levels and therefore does not need to run separate EAs for different complexity levels. This is even more important when the model incorporates far more than just three different complexity levels. The only challenge here is to find a balanced complexification chance, high enough to drive the EA population towards more complexity to expand the search to more areas on the one hand and low enough to allow for a thorough exploration of low complexity configurations on the other hand.

fixed EA complexity level	highest fitness	average fitness
1	6554	421
2	297926	81055
3	337406	158102

Table 5: Gathering Food Scenario: Best fitness results of the EA without complexification

The best solutions the EA with fixed complexity levels could find, of two and three species respectively, were nearly identical. They basically divided the bees into a forager and a defender group with a ratio of 1/3 being defenders and 2/3 foragers. Even within the individuals that contained three species, the third one was limited to one or none bees, which leaves it's influence negligible. In contrast, the complexification showed a wider variation in behavior. The third species, while still a minority, were able to reach up to 10% of the populations share and had thereby more influence on the behavior as a whole. Furthermore, defending bees were not only

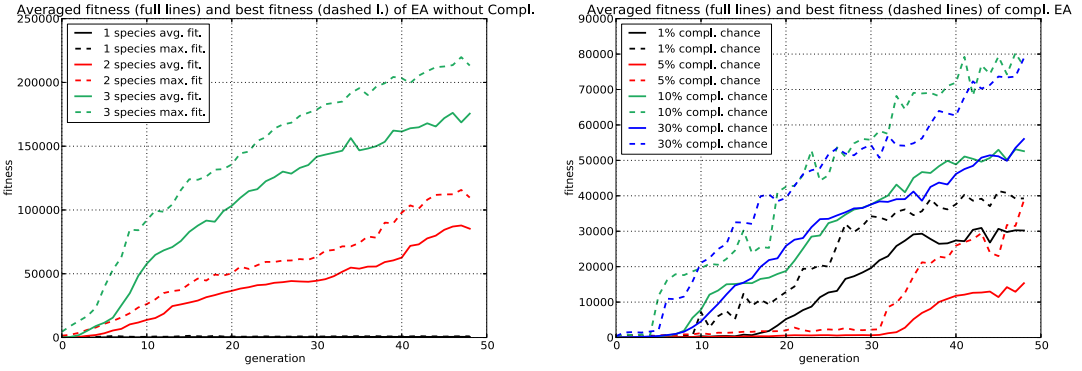

Figure 7: Gathering Food Scenario: left - survival scenario with standard EA, right - with complexification

complexification chance	highest fitness	complexity	average fitness
1 %	305547	2	28116
5 %	245079	2	17551
10 %	261772	3	42588
30 %	291573	3	48197

Table 6: Gathering Food Scenario: Best fitness results of the complexification runs

wandering around the proximity of the hive to intercept possible opponents, as they did in the fixed complexity EA runs, but could also stay put at a certain position to act as sentinels rather than moving guards.

6 Discussion and Future Work

We conducted experiments on two multi-agent based models to examine the effects of complexification. Both models benefited from its use, be it in the increase of fitness or a higher diversity in the resulting emergent behavior. The experiments in our work demonstrate that complexification improves evolutionary algorithms when it comes to genomes with variable length and complexity. Due to its adaptive nature it can explore a search space more dynamically by purposefully altering the individual's gene complexity to respond to the challenges of the problem at hand. A challenge here is to find a suitable complexification chance for the individuals as it can have considerable effects on the algorithm. It is an issue of balancing a too high and too low complexification pressure. In order to further guide the complexification and enhance its performance, several measures can be taken. One of them is simplification, a reverse of complexification. It allows individuals to decrease their complexity levels, in case that more simple genomes have higher chances of success. This can also alleviate a very high complexification pressure. Another enhancement can be pattern producing networks [8]. Rather than having each phenotype represented by one or more genes, this approach favors a mapping between genotype and phenotype, which is also more true to evolution in nature. Complexification would work as an alteration of the network that represents the mapping. The challenge of this

approach is that first a meaningful mapping network has to be found. It unfolds its benefits best when used in cases where there are symmetries in the phenotype and genes can be reused in order to create them. Furthermore it can considerably lower the genome size and therefore the potential search space.

7 Acknowledgements

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

References

[1] Sean Luke, Gabriel Catalin Balan, and Liviu Panait. MASON: A Java multi-agent simulation library. In *Proceedings of the Second International Workshop on the Mathematics and Algorithms of Social Insects*, Atlanta, Georgia, 2003.

[2] Sean Luke, Liviu Panait, Gabriel Balan, and Et. ECJ 16: A Java-based Evolutionary Computation Research System, 2007.

[3] M. Lynch and J. S. Conery. The origins of genome complexity. *Science*, 302(5649):1401–4+, 2003.

[4] Andrew P. Martin. Increasing Genomic Complexity by Gene Duplication and the Origin of Vertebrates. The American Naturalist. *Spring*, 154(2), 1999.

[5] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, ACM, pages 25–34, New York, NY, USA, 1987.

[6] Lee Spector, Jon Klein, Chris Perry, and Mark Feinstein. Emergence of collective behavior in evolving populations of flying agents. In *GECCO*, pages 61–73, 2003.

[7] Hari Sridhar, Guy Beauchamp, and Kartik Shanker. Why do birds participate in mixed-species foraging flocks? a large-scale synthesis. *Animal Behaviour*, 78(2):337 – 347, 2009.

[8] Kenneth O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, June 2007.

[9] Kenneth O. Stanley and Risto Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[10] Adam Stanton and Alastair Channon. Heterogeneous complexification strategies robustly outperform homogeneous strategies for incremental evolution. In *Advances in Artificial Life, ECAL 2013*, pages 973–980. MIT Press, September 2013.

[11] Forrest Stonedahl and Uri Wilensky. Finding forms of flocking: Evolutionary search in abm parameter-spaces. In *MABS*, pages 61–75, 2010.

[12] Michael Wagner, Wentong Cai, and Michael Harold Lees. Emergence by Strategy: Flocking Boids and their Fitness in Relation to Model Complexity. In *Proceedings of the annual Wintersim Conference 2013*, 2013.

[13] Kevin Winner, Don Miner, and Marie desJardins. Controlling particle swarm optimization with learned parameters. In *SASO*, pages 288–290, 2009.