Contents lists available at ScienceDirect
Journal of Computational Science
journal homepage: www.elsevier.com/locate/jocs

Evolving agent-based models using self-adaptive complexification

Michael Wagner[a,*], Wentong Cai[b], Michael H. Lees[c,d], Heiko Aydt[a]

[a] TUM CREATE, 1 Create Way, Singapore 138602, Singapore
[b] School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore
[c] Computational Science, University of Amsterdam, Netherlands
[d] ITMO University, St. Petersburg, Russian Federation

ARTICLE INFO

Article history:
Received 13 October 2014
Received in revised form 27 February 2015
Accepted 20 March 2015
Available online xxx

Keywords:
Evolutionary computing
Agent-based simulation
Modeling
Artificial life

ABSTRACT

This paper focuses on parameter search for multi-agent based models using evolutionary algorithms. Large numbers and variable dimensions of parameters require an optimization 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. To further improve performance we suggest and experiment with methods of self-adaptation to enable the algorithm to adjust its parameters individually to the problem at hand. We examined the effects of these methods on an EA by evolving parameters for two multi-agent based models.

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 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. Extending the EA to make use of complexification comes at the cost of introducing more parameters. To mitigate this we propose and

* Corresponding author. Tel.: +65 98393967.

http://dx.doi.org/10.1016/j.jocs.2015.03.005
1877-7503/© 2015 Elsevier B.V. All rights reserved.

experiment on mechanisms for self-adaptation of the additional parameter by the EA itself. 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 [6,13], 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 its 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.

Our 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 Eq. (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 Eq. (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.

In 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.

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

Fitness function of bee swarm scenario, relation between gathered and spent food.

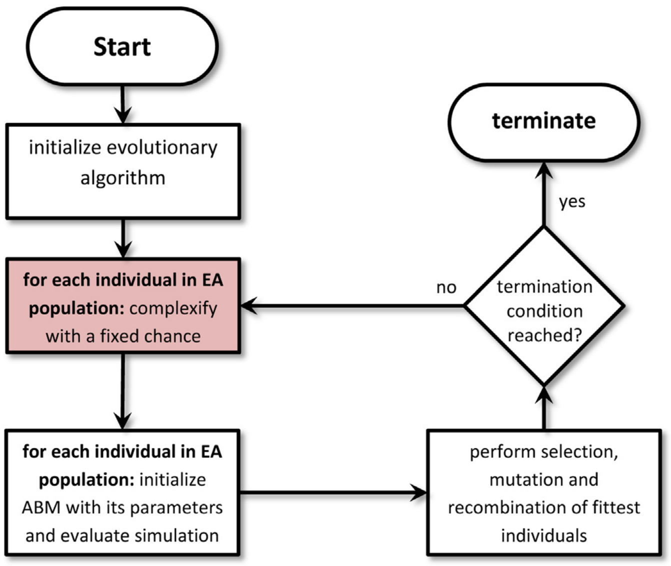
Start
initialize evolutionary algorithm
for each individual in EA population: complexify with a fixed chance
for each individual in EA population: initialize ABM with its parameters and evaluate simulation
perform selection, mutation and recombination of fittest individuals
termination condition reached?
no
yes
terminate

Fig. 1. Scheme of evolutionary algorithm, entry point of complexification highlighted.

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 first option directly translates into adding more possible combinations of genes. The latter option means to not 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 minimum requirement for integrating complexification into an EA is a mechanism to extend the genome (see Fig. 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 using 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.

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 vector as chromosome, which encodes the parameters for up to three groups of boid or bee agents respectively. In this case we can immediately define that a chromosome of minimal size has to have only one full set of parameters to completely specify the behavior for a single boid or bee species. As depicted in Fig. 2 the EA can then complexify a chromosome by duplicating the array representation of its genes and appending the copy to itself. The chromosome now has doubled its size which means that it encodes two species with identical parameters. This guarantees that the newly extended chromosome has a positive fitness and a valid configuration. Under the continued influence of mutation and crossover both parameter sets will slowly diverge in their values, leading to differing behavior of the respective species. The perpetual extension of genomes implies, for the GA, an increased coverage of the search space without changing the number of individuals or generations. 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, in the following referred to as P_{comp}, of 1%, 5%, 10% and 30% per individual per generation to compare the effects of different probabilities. Finally, the crossover has to be altered as well, 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 be crossed only 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 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

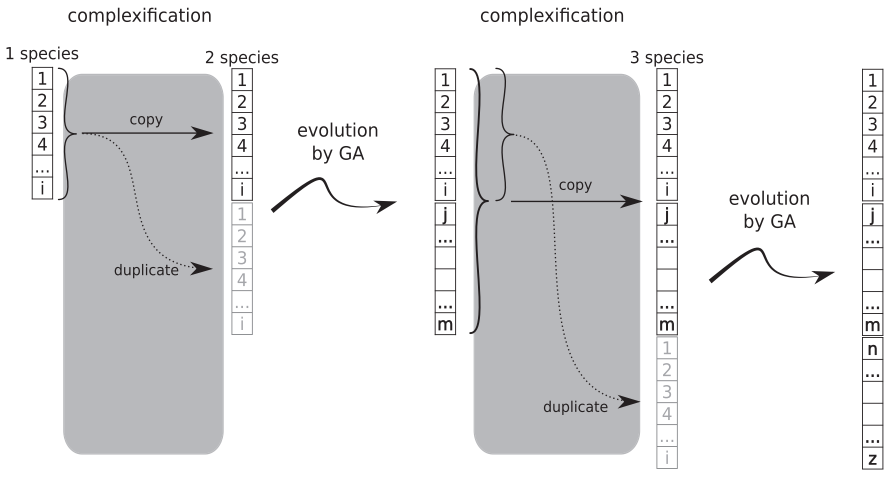

Fig. 2. Complexification of an EA individual, consisting of parameter vectors as chromosomes.

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 genes, only the necessary parts of the genotype will actually be used and evaluated, as seen in Fig. 2.

5. Experiments and results

In the following we will discuss the three 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 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 effectively and produce multi-species solutions with at least the same fitness as single species solutions.

5.1.1. 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 Fig. 3, with a sample size of 250,000 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

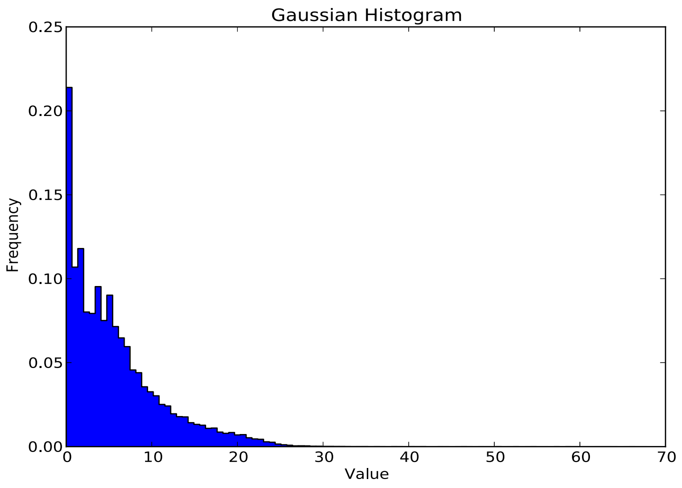

Fig. 3. Monte Carlo search on the survival scenario.

Table 1
Survival scenario: average and best fitness results of random search.

Complexity level	Average fitness	Highest fitness
1	1.33	67.8
2	3.03	40.48
3	1.44	28.6

Table 2
Survival scenario: average and best fitness results of the EA without complexification values averaged over 5 EA runs.

Fixed complexity level	Highest fitness	Avg. fitness
1	39.4	24.2
2	29.0	20.8
3	31.0	20.7

Table 3
Survival scenario: average and best fitness results of the complexification runs values averaged over 5 EA runs.

P_{comp} (%)	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

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.

5.1.2. Evolutionary algorithms

In our experiments we are using four different complexification probabilities. These indicate the chance for an individual to increase its complexity every generation. The chances we used for P_{comp} are: 1%, 5%, 10% and 30%. Higher values accelerate the complexification process and push the EA population faster toward 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 (Fig. 4).

The comparison between the two configurations, as seen in Tables 2 and 3, 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 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 Fig. 5 we can see how a high value for P_{comp} 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,

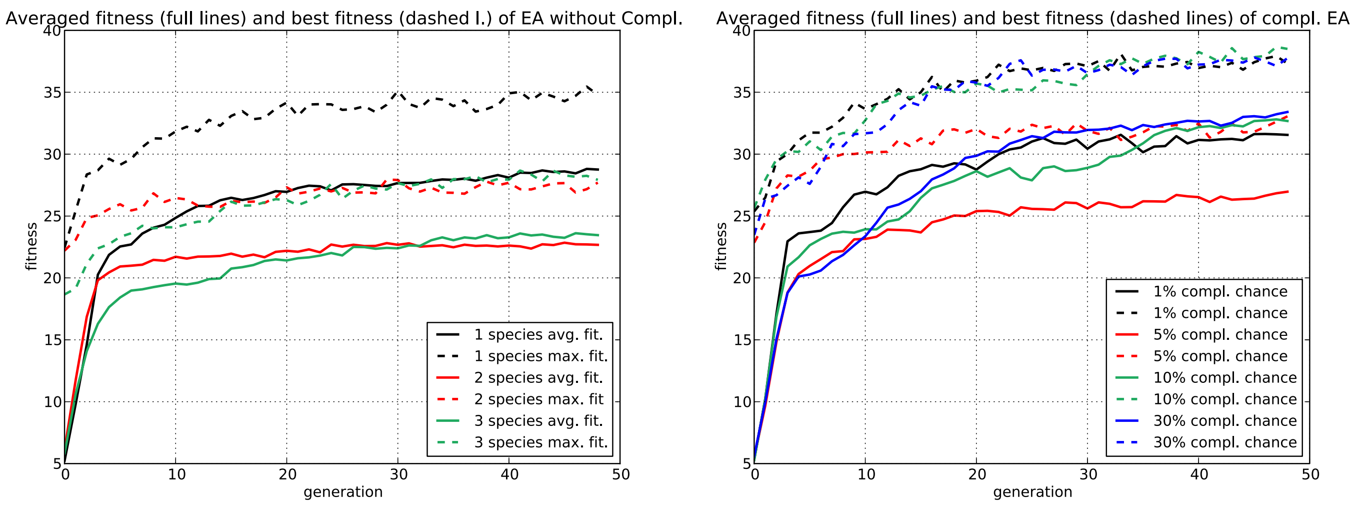

Fig. 4. Left: survival scenario without complexification, right: with complexification dotted lines are standard deviation.

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.

5.2.1. 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 200,000 samples indicates that the fitness landscape for the most part consists of very low-fitness individuals. The parameter configurations of the samples suggest that the search space is symmetrical. That means many solutions result in equivalent configurations where only the order is different in which the parameters are encoded. As an example, a solution with the configuration: 1st species – defender, 2nd species – forager corresponds to the solution: 1st species – forager, 2nd species – defender because they are the same. This indicates that the search space might have 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 135,856 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 Fig. 6. Note that in the inner graph 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.

Table 4
Gathering food scenario: average and best fitness results of random search, values averaged over 5 EA runs.

Complexity level	Average fitness	Highest fitness
1	4.01	8762
2	64.27	135,856
3	26.81	125,523

5.2.2. 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

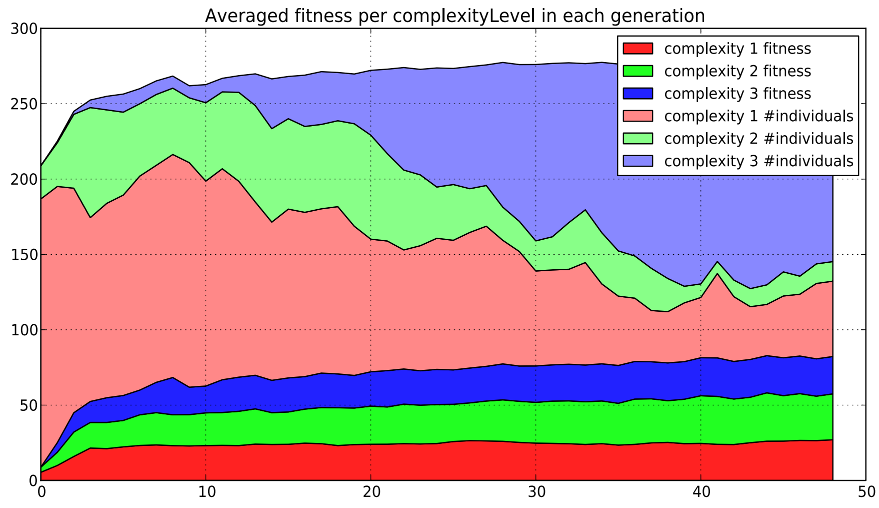

Fig. 5. Survival scenario: Light colors – relationship between number of individuals with respective complexity level. Dark colors – relationship between fitness of individuals with respective complexity level.

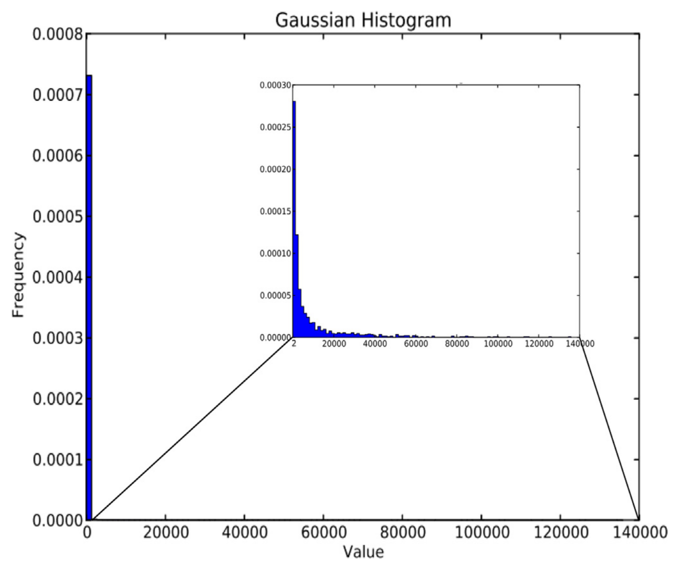

Fig. 6. Random search on the gathering food scenario. The inner diagram starts at $x = 2$ and has a different scaling to show the huge difference in numbers between individuals with fitness of 0 or 1 and individuals with higher fitness.

Table 6
Gathering food scenario: best fitness results of the complexification runs, values averaged over 5 EA runs.

P_{comp} (%)	Highest fitness	Complexity	Average fitness
1	305,547	2	28,116
5	245,079	2	17,551
10	261,772	3	42,588
30	291,573	3	48,197

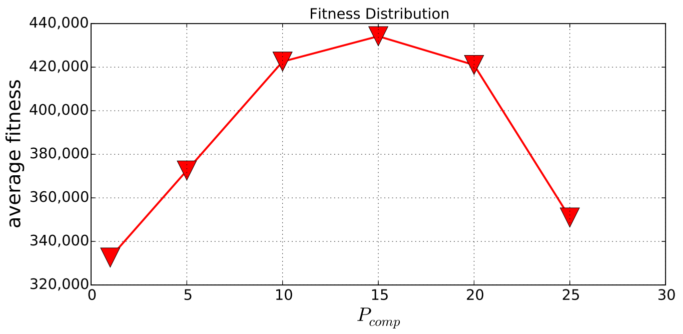

Fig. 8. Highest fitness results for a series of P_{comp} values.

Table 5
Gathering food scenario: best fitness results of the EA without complexification, values averaged over 5 EA runs.

Fixed EA complexity level	Highest fitness	Average fitness
1	6554	421
2	297,926	81,055
3	337,406	158,102

the average over the repetitions. Similar to the survival experiment, we again use EAs with four different P_{comp}. 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 Fig. 7), the highest fitnesses 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 values for P_{comp}, high enough to drive the EA population toward more complexity to expand the search to more areas on one hand and low enough to allow for a thorough exploration of low complexity configurations on the other hand.

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 is 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 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. Fig. 8 shows a range of values for P_{comp} with the highest fitness achieved using each of them. It exhibits close resemblance to a Gaussian distribution, which indicates the existence of an optimal complexification probability. This sweet spot has P_{comp} that satisfies two criteria. First, P_{comp} has to be large enough to explore the search space within a give time limit, but still small enough to not degenerate the evolution into a random search.

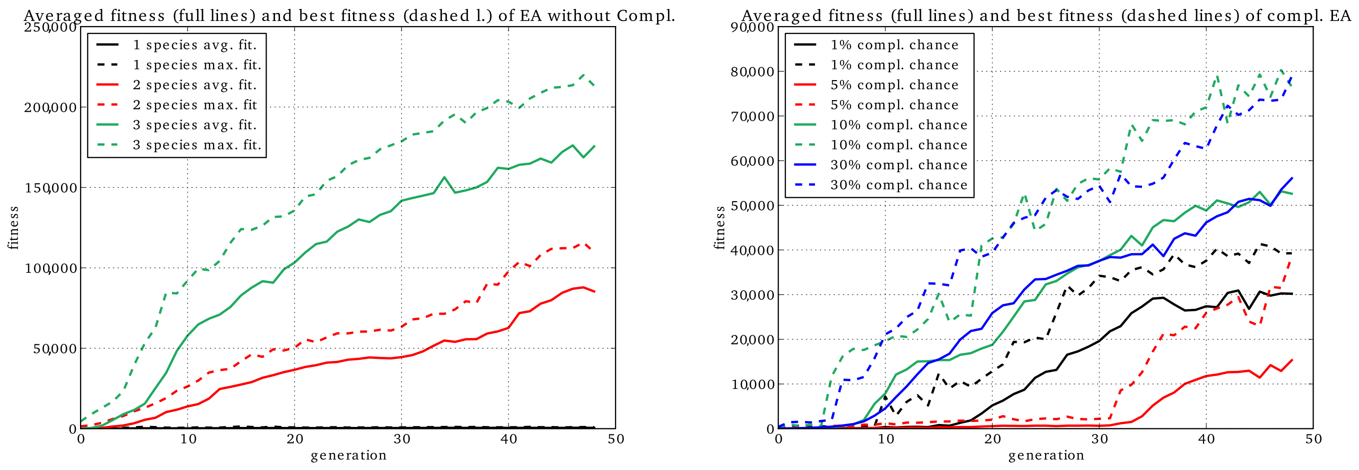

Fig. 7. Gathering food scenario: left – with standard EA, right – with complexification.

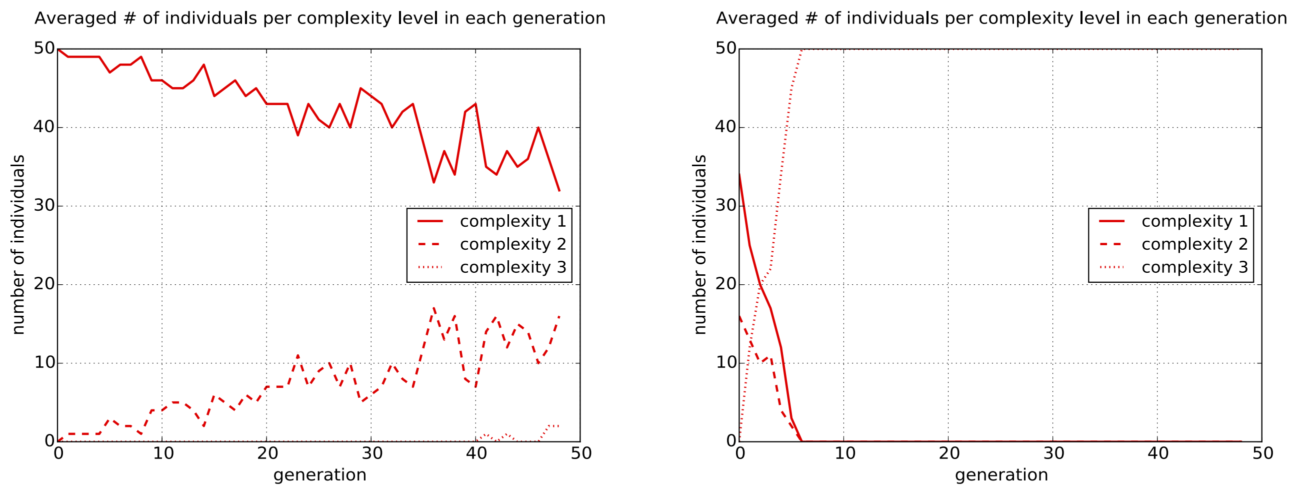

Fig. 9. Adaptive complexification experiments of gathering food scenario: left – with linear increase of P_{compl}, right – with evolving P_{compl} values averaged over 5 EA runs.

5.3. *Automatic adaptation of the complexification rate*

Our previous experiments, the bee swarm model in particular, indicate that convergence and fitness results of the complexification EA are greatly influenced by the value of P_{comp}. A low probability provides ample time for optimization within the initial search space while higher rates benefit a faster expansion into more complex solutions. To avoid having to individually tailor P_{comp} to every new problem, we want to explore methods of automatic adaptation. One possibility is to add the rate of the chromosome as a new parameter which is also evolved by EA. A side effect to this is that now P_{comp} is not identical for all individuals anymore, but uniquely fitted to each of them. It is not initially clear whether this is beneficial or detrimental to the optimization process.

5.3.1. *Experiments*

For the following experiments we use the bee swarm model again as it offers more room for improvement than the boid model optimization. The latter already achieves results close to the global optimum, given that no more boids can survive than there are initially in the scenario whereas bee swarm model has no predefined maximum value of gathered food. The first experiment is designed to linearly increase P_{comp} to obtain a benchmark that we can compare with the second experiment. The linear increase starts with P_{comp} set to 0.01 and increases by 0.03 every five generations for all individuals at the same time. At the end of running for 50 generations it will reach 0.28. All other parameters are assumed to use the values we previously tested in separate runs in Section 5.2. In the second experiment, P_{comp} is included in the chromosomes of the EA and evolved alongside all other parameters. At EA initialization P_{comp} is randomly set for each chromosome, it can assume values between 0 and 1 and is subjected to Gaussian mutation with a standard deviation of 0.05. All other simulation parameters will remain the same as for the previous experiments.

In the following we compare the results of both methods. The linear increase of P_{comp} causes a gradual rise of EA chromosomes with higher complexity, as seen in Fig. 9. Though P_{comp} reaches values as high as 28% probability of complexification per individual per generation, the EA finishes with populations that have, on average, still 60% of individuals with complexity level of 1. The other chromosomes add up to about 35% with complexity level 2 and 5% with complexity level 3. As is visible in the Fig. 9, there are still a lot of low complexity individuals because the high complexification probabilities for P_{compl} are only reached in the final generations of the evolutionary algorithm. In contrast, the inclusion of P_{comp} into the evolution exhibited a very strong convergence toward a high complexification pressure, and in consequence, causing a large number of solutions of higher complexity as soon as after only five to six generations.

If we look at how the average fitness develops during the experiments, as depicted in Fig. 10, we see that the linear increase method achieves an average fitness of close to 225,000 units of gathered

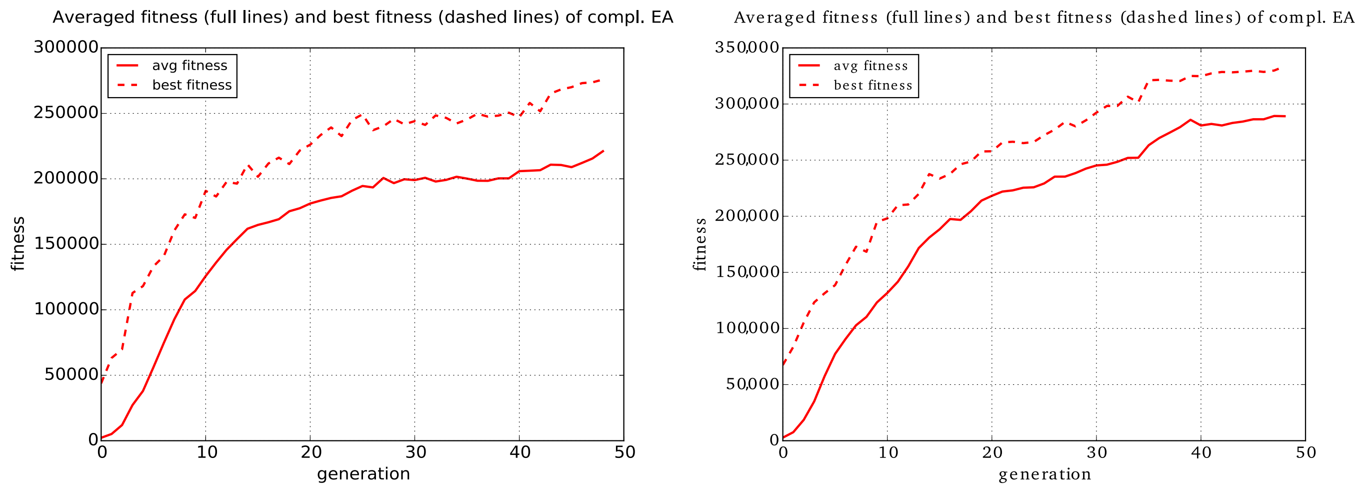

Fig. 10. Adaptive complexification experiments of gathering food scenario: left – linear increase, right – evolution of P_{comp} values averaged over 5 EA runs

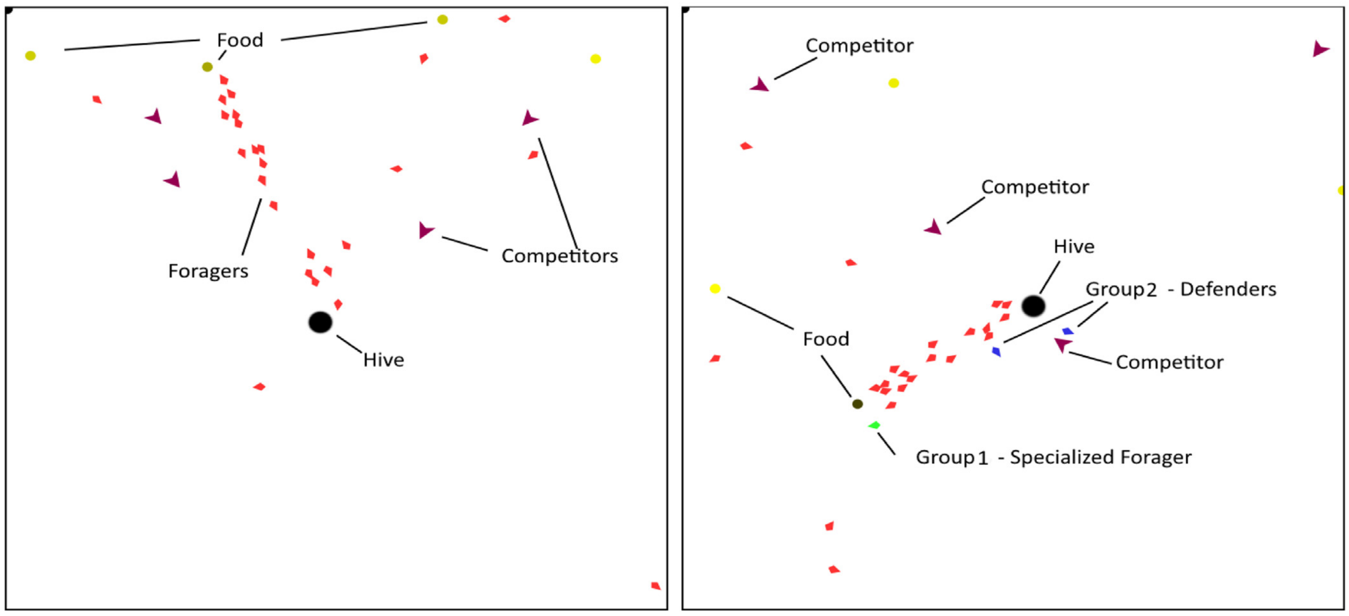

Fig. 11. Simulation screenshots: left – one species configuration found by linear increase, right – three-species configuration found by evolutionary method values averaged over 5 EA runs. (For interpretation of the references to color in the text, the reader is referred to the web version of this article.)

food. That amount considerably exceeds the results of each of the previous experiments with fixed values for P_{comp} of 1%, 5%, 10% and 30%. It furthermore suggests that dynamic complexification probabilities are evidently more suitable for efficiently extending the search space, rather than fixed values, which do not change over the course of the optimization, or no complexification at all. By comparing linear increase with the results of the evolution of P_{comp} we find that the evolutionary method improves the optimization results even further, see Fig. 7. The amount of food gathered reaches as high as 280,000 units on average. It is also interesting to note that linear increase converges after about half of all generations passed. It then only improves marginally whereas the evolutionary method exhibits a more steady increase of the fitness until it converges in the last ten generations.

The differences between the solutions found by linear increase and evolutionary method become apparent when we take a look at the resulting simulation configurations. One of the strongest configurations found by linear increase consists of one bee species, entirely specialized on foraging. This strong focus on pure maximization of foraging gain allows the bees to compensate the losses incurred by the competitors, as seen in Fig. 11 on the left side. On the other hand the solution by the evolutionary method does not differ very much from the ones found by the previous complexification experiments with static values for P_{compl}. However it manages to yet optimize the parameters better. In Fig. 11 on the right side we again see three species where the largest group (shown in red) is specialized on foraging and scouting, to capitalize on the movement speed which is proportional to scouting skill, while only having extremely rudimentary defending capabilities. The much smaller second group (green) with only a single bee entirely specializes on foraging with no defending skills at all whereas the last group (blue) purely serves as highly skilled defenders, with a weaker specialization in scouting benefit from the movement speed bonus again. One

Table 7
Gathering food scenario: best fitness results of the complexification runs.

Method	Highest fitness	Complexity	Average fitness
Linear increase	350,484	1	221,000
Evolving P_{compl}	389,212	3	281,000

interesting observation is that the scouting skill was never used to primarily specialize in but, if at all, only to enhance either foraging or defending capabilities by making use of the increased movement speed. This may be caused by the world size which is evidently small enough for the bees to not require any dedicated scouts for finding food.

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, either 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 the use of genomes with variable length and complexity is possible. Due to its adaptive nature it can explore a search space more dynamically by starting off with simple chromosomes, and then gradually increasing the chromosome size to extend the search space and respond to the challenges of the problem at hand. Another finding in this work is that the value of the complexification probability P_{comp} can have a big impact on quality of the optimization result. A probability that is too high may discard simple solutions too easily by pushing too fast toward increased chromosome sizes. The inverse case happens when the probability is too low, most chromosomes remain rather short in length which renders the algorithm unable to find any global optima that require more complex solutions and longer chromosomes to find. We also showed that adaptive techniques are a sensible tool for letting the evolutionary algorithm find the optimal complexification rate automatically to remedy the disadvantages of fixed complexification rates. Apart from evolving the value for P_{compl}, another possible way for future research is to use a metric that uses a relationship between complexity of chromosomes and their fitness to determine whether a higher or lower complexification chance is needed to improve convergence toward the desired fitness values. A third option is to use the 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.

Acknowledgements

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

Michael H. Lees is partially supported by Russian Scientific Foundation, Project #14-21-**00137**.

References

[1] S. Luke, G.C. Balan, L. 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] S. Luke, L. Panait, G. Balan, et al., ECJ 16: A Java-based Evolutionary Computation Research System, 2007.
[3] M. Lynch, J.S. Conery, The origins of genome complexity, Science 302 (5649) (2003) 1401–1404.
[4] A.P. Martin, Increasing Genomic Complexity by Gene Duplication and the Origin of Vertebrates, Am. Nat. Spring 154 (2) (1999).
[5] C.W. Reynolds, Flocks, herds and schools: a distributed behavioral model, in: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, ACM, New York, NY, USA, 1987, pp. 25–34.
[6] L. Spector, J. Klein, C. Perry, M. Feinstein, Emergence of collective behavior in evolving populations of flying agents, in: GECCO, vol. 6, 2003, pp. 1–73.
[7] H. Sridhar, G. Beauchamp, K. Shanker, Why do birds participate in mixed-species foraging flocks? A large-scale synthesis, Anim. Behav. 78 (2) (2009) 337–347.
[8] K.O. Stanley, Compositional pattern producing networks: a novel abstraction of development, Genet. Program. Evol. Mach. 8 (2) (2007) 131–162.
[9] K.O. Stanley, R. Miikkulainen, Competitive coevolution through evolutionary complexication, J. Artif. Intell. Res. 21 (2004) 63–100.
[10] A. Stanton, A. Channon, Heterogeneous complexication strategies robustly outperform homogeneous strategies for incremental evolution, in: Advances in Articial Life, ECAL 2013, MIT Press, 2013, September, pp. 973–980.
[11] F. Stonedahl, U. Wilensky, Finding forms of flocking: evolutionary search in ABM parameter-spaces, in: MABS, 2010, pp. 61–75.
[12] M. Wagner, W. Cai, M.H. Lees, Emergence by strategy: flocking boids and their fitness in relation to model complexity, in: Proceedings of the Annual Winter-sim Conference, 2013, 2013.
[13] K. Winner, D. Miner, M. Desjardins, Controlling particle swarm optimization with learned parameters, in: SASO, 2009, pp. 288–290.