Evolutionary Optimization of Societies in Simulated Multi-Agent Systems

Christoph Oechslein, Alexander Hörnlein, Franziska Klügl

University of Würzburg, Department for Artificial Intelligence,
{oechslein|hoernlein|kluegl}@informatik.uni-wuerzburg.de
http://ki.informatik.uni-wuerzburg.de/

Abstract In this paper we show how to use evolutionary optimization in multi-agent systems with simple, non deliberative agents to evolve successful interaction in societies, mostly for task allocation based problems.
The presented generic optimization scheme – which is implemented based on the multi-agent simulation tool SeSAm – was used to model prehistoric ant species. The idea was to show how solitary individuals evolve to societies. Results from these experiments are given. Finally future plans are discussed and evaluated.

Introduction

Simulation is a widely accepted tool to model systems and to use this modeled system for answering questions. These answers of the modeled artificial system can be used to predict the behaviour of the real system which is one way simulation models are used.

In multi-agent simulations many intelligent entities (agents) interact among each other and with a special entity, the world. Often the spatial relationship between the agents is an important feature in these models. An example is an ant colony, where the interacting ants are agents which cooperatively try to find and hunt down prey or to exploit other food sources.

Cooperation is especially important for multi-agent systems with simple, non deliberative agents. In their intelligent and massive interacting lies the power: 'The whole system is more than the sum of its parts'. Because modeling these 'organizations' – which are in our understanding more 'task allocation' problems – is a rather complex task, an idea is to use an automated method to find good organization structures for a given problem. Based on an evaluation function, this process can be transformed into an optimization process, so the process of determining organizational structures can be solved by a generic optimization component.

In the next section a few approaches to include optimization in a simulated multi-agent system are sketched. After that the details of the implementation and evaluation experiments are given.

Optimization in Simulated Multi-Agent Systems

In the literature the optimization of simulated multi-agent systems is hardly handled, optimization is often just seen in the context of general simulation, as e.g. in [8]. These generic approaches can also be used for optimizing multi-agent simulations. One simulation run with certain variable settings is seen as one point in the search space with an associated evaluation value (response-surface of the system). In this space 'normal' search (like Hill-climbing) or evolutionary inspired algorithms [8, 2] try to find optimal (good) values.

In these approaches a simulation run is treated as a black box. This view has drawbacks because often the specific simulation experiments could be terminated due to foreseeable bad evaluation value. Such a termination depends on the evaluation function which is – in these approaches – not available to the simulator. This kind of early termination is only applicable if the evaluation function is not defined for a complete simulation run or if there are additional heuristics available.

In our opinion this early termination is crucially for using optimization in multi-agent systems, mainly due to the very time-consuming simulation runs (several hours). Another reason is the often non-deterministic behaviour which requires few runs to get statistically concrete predications. On the other hand the evaluation criterium is mostly based on single agents or on an aggregate function of individual fitness, thus determining the time for an early termination is easily possible.

But there are also other approaches using optimization in multi-agent systems. Sometimes we want to improve the performance of a single agent (or a group of agents) so one could optimize *during* the simulation run. This is – as mentioned before – not possible for all evaluation functions. The optimization of an individual single agent is possible at different levels, if the behaviour of the agents changes during its lifetime, it is learning, if agents can reproduce and the offsprings are different, it is evolution.

Our goal for the additional optimization component to our simulation modeling tool SeSAm was to give the modeler of simple non deliberative agents an easy tool to optimize. Mostly the complex interaction needed in these societies are an optimization goal. But before presenting concrete details of our optimization solution the basic simulation environment is sketched.

SeSAm

The SeSAm (ShEll for Simulated Agent SysteMs, see [6]) provides a generic environment for modeling and experimenting with agent-based systems. We specially focused on providing a framework for the easy graphical construction of complex models.

As a graphical interface for implementing a model, built-in animation capabilities, tools for collecting and analyzing protocol data, etc. are provided, even scientists without traditional programming experience are able to build and experiment with multi-agent models. In Fig. 1 the structure of the environment is presented. In principle all of the provided tools are domain-independent, but in order to facilitate modeling the system can be adapted to a given domain.

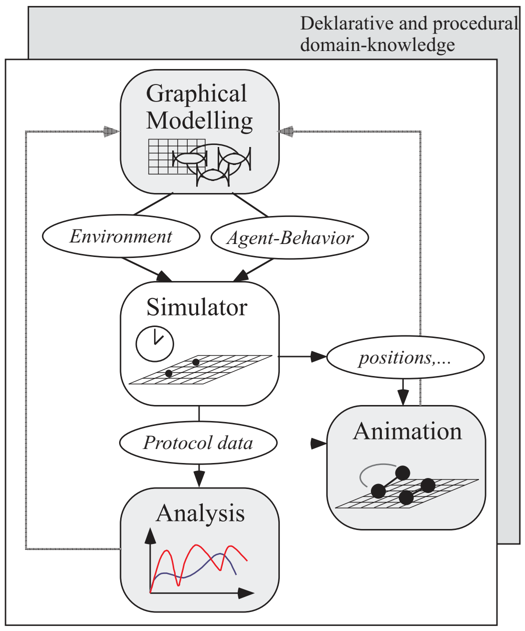

Figure 1: The SeSAm Simulation environment.

An agent is specified by the following three categories:

1. Sensoric abilities and internal variables,
2. the action selection procedure including all internal representation used for it and
3. the effectoric abilities.

The most important points for this paper is the action selection architecture. Each agent has a current activity which determines the actions it executes on itself and on its environment once every time step. The termination of the current activity is driven by specified termination rules. The rule based transitions between different activities and the activities themselves can be formulated like every other part of the behavior model using tools as depicted in Fig. 2 and forms an activity graph as shown in Fig. 4.

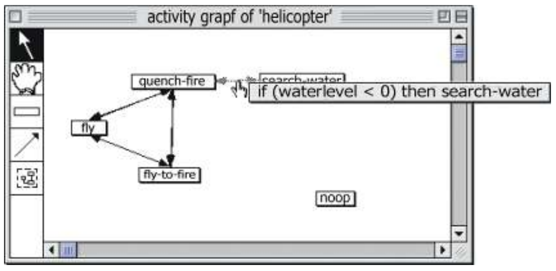
activity grapf of 'helicopter'
quench-fire
search-water
if (waterlevel < 0) then search-water
fly
fly-to-fire
noop

Figure 2: Graphical representation of activities and transitions can be used as a powerful tool for editing and navigating.

As can be seen in the last figures rules can be parameterized by variables of the agents, which plays an important rule in the evolution component described in the next section.

The Evolution Component

As mentioned above optimization is crucial to the design process of multi-agent systems, since the exact – quantitativ – interactions between agents are hard to specify.

The purpose of this component was to provide a generic tool to help modelers run their own evolution experiments, and to let the evolution find quantitative relationships. The first goal is particularly important for us as we are cooperating with researchers in biology. These experts use SeSAm for building models, to answer questions, like 'How did evolution manage to find organizational structures as found in ant colonies?'.

Back to the design an important question is how the search space of the genetic search should be structured. There are mainly two distinct approaches: Using some kind of atomic values to parameterize a given solution or find the whole solution. The first approach is the most common in GA[3] or ES[1], the second one is used by GP[7]. To optimize societies in the first approach the modeler would build a parameterized activity graph where evolution tries to find good parameter settings. In the second approach the modeler would only specify what action and sensors the agent could have and the evolution tries to find the whole activity graph.

GP evolution has the advantage of having the possibility to find all solutions to a problem, but that's also the handicap of this approach since it needs a lot of resources (computer time and space) to evaluate solution and the resulting solution is often hard to understand.

The implemented evolution component operates on numbers like ES, i.e. one gene value is a number which can be used in the activity graph, e.g. in transition rules (`if ([gen value 1] > 0.5)` $\rightarrow$ `queen-task`). So the organizational structure – mainly the task allocation structure – is first encoded by the modeler in the parameterized activity graph and the evolution tries to find good parameter settings.

There exist a few recombination operators and a few mutation possibilities which are explained in the following sections. The calculation of the fitness is by 'surving of the fittest', i.e. an agent can mate and die either is responsible to the modeler. She has to incorporate these actions in the model (as you can see in Fig. 4). The modeler could also implement a combination of fitness calculation and 'surving of the fittest' by modeling more selective choice of the mating partner.

Mutation in biology and also in computer science is the local search part of the evolution. Therefore the mutation operator should not change the gen 'too much'. This is accomplished allowing smaller changes with higher probabilities and larger ones with less, i.e. the mutationmodification is distributed using a normal probability function as shown in Fig. 3. The standard deviation of this normal distribution is also a part of the genom of an agent as in standard, 'modern' ES strategies (and also in other see e.g. [4] for a survey) allowing the evolution to find the appropriate step size for mutation.

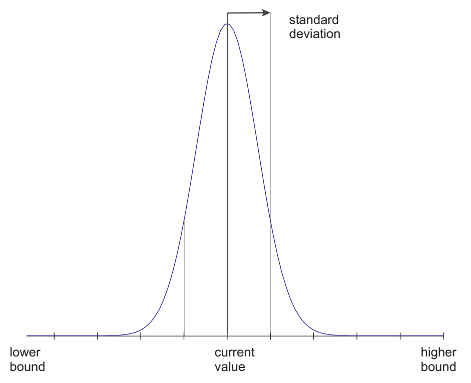

Figure 3: Distribution for mutating the value of a gen.

With this component a few scenarios were implemented, the most important and broadest is described in the following. More about the experiments can be found in [5].

The Ant Experiment

In this scenario the agents were given a broad repertoire of behaviours. Each agent could 'hunt', 'mate', 'fight', 'take care of brood' but the preference to the several behaviours is given by a few inherited variables. Among other genes like 'when to start laying eggs' or 'at what level of energy eating brood', the most important genes are the following (also look at Fig. 4):

queen factor: How often is the agent in this part of the activity graph.
soldier factor: How often the ant leaves the nest to hunt and how good is it at fighting.
nurse factor: Determines when to take care of brood.

Since being good at all things always has advantages, the disadvantages of being an allrounder is the added complexity in the body/brain with is modeled by needing more energy to construct such an offspring. Each factor has a lower bound – ants with values below are completely unable to perform the corresponding behaviour.

Experimenting with this model showed the emergence of spots which some affiliated ants. In these 'nests' also occur specialization: There exist plain queens and plain non queens. In Fig. 5 the timeline of the change in the queen, soldier and nurse factor is given. The queen factor tends to the lower bound, which means that only 10% of offsprings of a queen are itself queens. The nurse and hunt factor stay at the higher bound, meaning all ants can take care of brood and go hunting. In other runs we noticed the correlation between low hunt factor and high distance between nests or high pray density. Also if the pray density is high the nurse factor declines.

These simulation runs showed, that under particular circumstances it is more profitable for individual ants to not engage in reproduction and instead to work for the colony. More experiments and the tighter inclusion of biologists should validate the model and give us the chance to reason better about the results.

Results & Discussion

Modeling societies in multi-agent systems is hard (but necessary) – particularly with simple, non deliberative agents –, since the interactions between the agents are complicated. Our approach is to use optimization to automatically build successful societies. We explained the generic method used in 'normal' simulation for optimization and argued – mainly due to the long duration of one simulation run in multi-agent systems –, that an other approach could be more fruitful.

We introduced shortly our multi-agent simulation tool SeSAm, in this tool it is possible for the domain expert to model graphically. Afterwards we explained the added evolution component, which enable individual optimization *during* a simulation run. We use a ES evolution method with 'survival of the fittest' as selection strategie. It tries to optimize a given parameterized activity graph.

The component was tested and evaluated in a biological inspired model, where the evolution from indivual 'ants' to colonies of ant with division of labor emerged. The tests was very promising and our cooperation with researchers in biology will be fruitful. Also other kinds of optimization levels in multi-agent systems should be explored, like the generic, the learning or the static approach. Static approach would be optimization of a model without simulating.

References

[1] Bäck, Th. and H.-P. Schwefel: *Evolutionary computation: An overview.* In *Proceedings of the Third IEEE Conference on Evolutionary Computation*, pages 20–29. IEEE Press, Piscataway NJ, 1996.

[2] Glover, F., J. Kelly, and M. Laguna: *New advances and applications of combining simulation and optimization.* In Charnes, J., D. Morrice, D. Brunner, and J. Swain (editors): *Proceedings of the Winter Simulation conference*, pages 144–152, 1996.

[3] Goldberg, D.: *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Mass., 1989.

[4] Hinterding, R., Z. Michalewicz, and A.E. Eiben: *Adaptation in evolutionary computation - a survey.* In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation.*

[5] Hörnlein, A., C. Oechslein und F. Puppe: *Optimierung in simulierten biologischen Multiagentensystemen mit Hilfe evolutionärer Verfahren.* In: Klügl, F., F. Puppe, P. Schwarz und H. Szczerbicka (Herausgeber): *Bericht des Instituts für Informatik*, Band 253, 2000.

[6] Klügl, F. and F. Puppe: *The Multi-Agent Simulation Environment* SeSAm. In Büning, H. Kleine (editor): *Proc. of the Workshop Simulation and Knowledge-based Systems 1998*, 1998.

[7] Koza: *Genetic Programming III.* Morgan Kaufmann Publishers, Inc., San Francisco, 3rd edition, 1999.

[8] Law, A. and D. Kelton: *Simulation Modelling and Analysis.* McGraw-Hill, 3rd edition, 2000.

[9] Oechslein, C., F. Klügl und F. Puppe: *Kalibrierung von Mutiagentenmodellen.* In: Köchel, P. (Herausgeber): *KI-Methoden in der simulationsbasierten Optimierung (CSR-99-03)*, Seiten 71–78. Chemnitzer Informatik-Berichte, 1999.

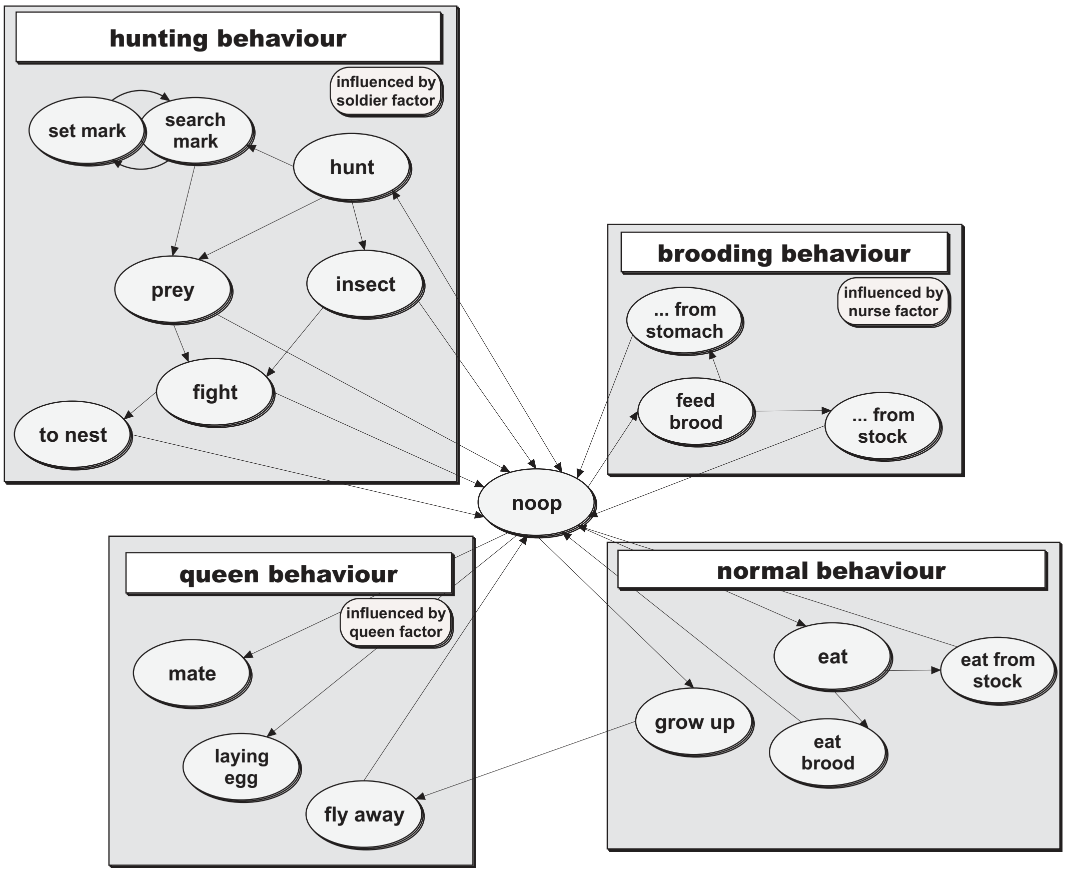

Figure 4: Graph of possible behaviours of prehistoric ant.

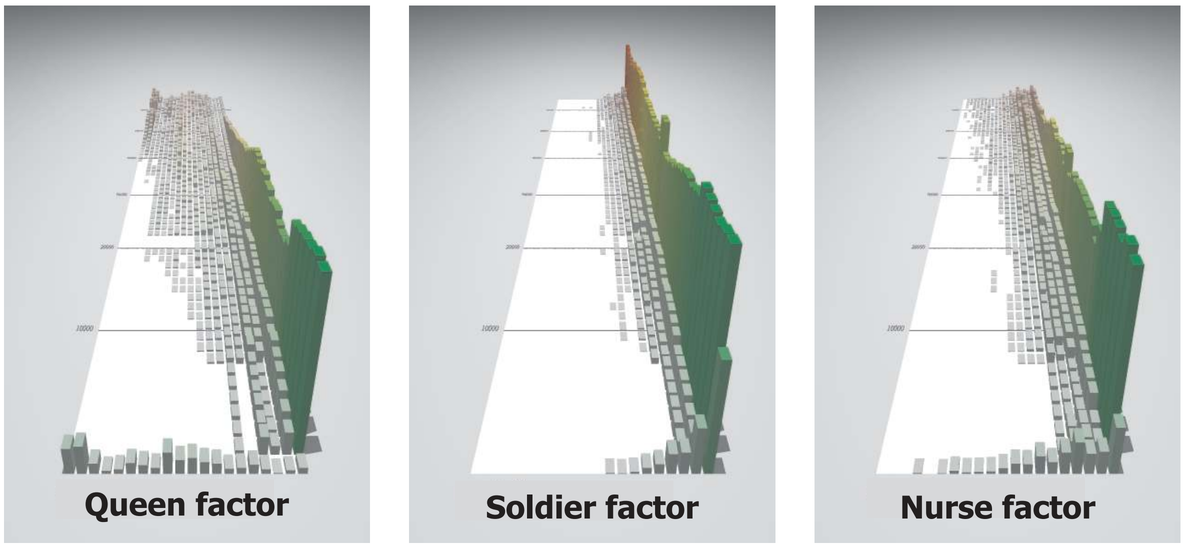

Figure 5: Change in genpool: The x-axis is the value of the gen, the y-axis is the frequency of the values and the z-axis is the time. The first row is the last result of this run.