An Evolutionary Approach to Grid Computing Agents

Yvonne Bernard, Lukas Klejnowski, David Bluhm, Jörg Hähner, Christian Müller-Schloer

Institut für Systems Engineering, FG System- und Rechnerarchitektur
Leibniz Universität Hannover, Appelstrasse 4, D-30167 Hannover
{bernard, klejnowski, bluhm, haehner, cms}@sra.uni-hannover.de

Abstract. The Organic Computing initiative aims at introducing new, self-organising algorithms in order to cope better with the complexity of today's systems. One approach to self-organisation is the introduction of agents which are able to continuously adapt their behaviour to changing environmental conditions and thus collectively create an efficient and robust system. In this paper, we introduce an evolutionary approach to an agent which acts autonomously in a Desktop Grid system. The evolutionary agent is defined by ten chromosomes defining its behaviour. If two agents interact, the inferior agent copies a part of the genes of the more successful agent. Therefore, the most successful gene combination will spread throughout the network.

1 Introduction

Organic Computing(OC) offers a variety of algorithms and mechanisms to enable cooperation between heterogeneous subsystems in large-scale, complex systems. For these highly dynamic systems, not all possible configurations can be foreseen at design time. Therefore, self-X properties, such as self-organisation, self-configuration or self-healing are used to optimise these complex systems at run-time and thus cope with their dynamics.

A new research focus in the OC context is Social OC [8], which transfers concepts and knowledge gained from social systems and institutional economics into system architectures. A project within this new research area is the OC-Trust project which aims at improving both the cooperation among subsystems and the robustness regarding malicious behaviour using trust-based algorithms. Ensuring the trustworthiness of subsystems will enable system designers to realise the openness of complex, highly dynamic systems, i.e. the dynamic inclusion of formerly unknown agents.

One approach to the management of complex, dynamic systems is the usage of Adaptive Agents [2]. These agents are able to fit their behaviour to the current situation they observe based on predefined thresholds. These thresholds are tailored to the situation, e.g. if there is a high workload, an agent needs to ask more (and occasionally even less trustworthy) agents for cooperation. For each situation the system designer defines a suited threshold, based on his knowledge of the system at design-time. However, we want these agents to learn and optimise at run-time the threshold best suited in a given situation. One possibility for optimisation is an evolutionary approach where during agent interaction, a new population arises, continuing life with the dominant genes of the successful agents from the last generation. This completely distributed way of learning and optimisation seems to be worthwhile considering for the agents in our application scenario. Therefore, in this paper, we introduce and evaluate a new class of agents called Evolutionary Agents which optimise the decision making in both worker and submitter role at run-time by imitation of the fitter agents in combination with mutation.

This paper is organised as follows: First, the application scenario Trusted Desktop Grid, which has been used for the evaluation in this paper, will be introduced and a short overview on related work is given. In Section 3, the design and implementation of our Evolutionary Agent will be given. We will evaluate how the Evolutionary Agents behave, both in a homogeneous system and in a heterogeneous system with Adaptive Agents, in Section 4. In the last section, we will conclude the paper and give an outlook on our future work.

2 Application scenario: Trusted Desktop Grid

The application scenario for our research is a desktop grid and volunteer computing system (DGVCS, [4]) with agents acting on behalf of the users. The system is designed as a distributed system without central control. Clients have the capabilities to be both submitters and workers, which is described below in more detail. The clients are assumed to be heterogeneous in terms of administrative domains, machine resources, usage patterns, volatility etc. Such a grid is suitable for scenarios where most clients run applications that produce grid jobs and thus are in high demand of computing resources.

In the Trusted Desktop Grid, agents become *submitters* whenever a user application on their machine produces a grid job. These jobs are split into single work units (WUs) which are distributed among available worker clients. The *workers* process them and return the results to the submitters which validate the results. However, these systems are exposed to threats by clients that plan to exploit or damage the system. A worker can for example return an incorrect result or not return a result at all. Workers can also refuse to accept a WU. Here, trust mechanisms can help the agents to estimate the future behaviour of other agents. By extending each client with an agent component and modelling the relations between the agents with a trust mechanism, we expect to counter these threats and thus increase the efficiency of such a system. If, for instance, an agent chooses only those workers that it already had good experiences with, the expected outcome is better. In this paper, the desktop system introduced above has been evaluated in a multi-agent simulation. Agents in this system can act as submitter (i.e. decide which worker to give WUs to) and worker (decide whose WUs to accept) at the same time. However, agents following static rules according to a fixed trust model can not succeed in a highly dynamic system. Volatile peers, changing trust relations, different workloads and user goals all require the agents to adapt in order to be successful. Moreover, we want the agents to autonomously decide between a more egoistic and a more altruistic behaviour and learn which behaivour is successful in a situation. In the Evolutionary Agent approach introduced in this paper, we show and evaluate how learning and run-time optimisation using an evolutionary approach can be used in our application scenario.

2.1 Related work

The idea of using evolutionary approaches for Grid Computing has, for instance, been introduced in [1]. The paper shows the development of genes in a simulated evolutionary peer-to-peer overlay scheme. In contrast to this, the behaviour of our evolutionary agent approach is not only evolutionary, but also based on trust. An overview of trust and reputation concepts can be found in [7]. Our agents are given trust and reputation values determined by former interactions. Thus, trust is used as a constitutional part of the agent cooperation relations as discussed in Section 2. Our Evolutionary Agent approach is based on the idea of using evolution to model trust relations in Multi-Agent Systems (MAS) as introduced in [6]. The authors of [6] have introduced a chromosome structure which is able to model trust-based behaviour of agents. With this model, trust creation, destruction and rebuilding can be realised and analysed using a graphical representation of the agents' genes. We have modified this model by creating a new chromosome structure tailored to the Grid agents in our application scenario Trusted Desktop Grid. We have simulated the Trusted Desktop Grid as MAS. A justification for the usage of agents for Desktop Grid systems is given in [5].

3 Evolutionary Agent

The Evolutionary Agent is an approach to run-time optimisation of trust-based interaction. Based on [6], we aimed at creating an evolutionary agent model, which enables cooperation and trust-building. This model has been adapted for our application scenario Trusted Desktop Grid in order to make it the basis of the worker and submitter decisions.

In this section, we will introduce the design of this agent type in general as well as how this design is used to make the agent decisions in the application scenario Trusted Desktop Grid in both worker and submitter role.

The Evolutionary Agent consists of a chromosome structure, which contains 10 genes. The bit values of the genes represent the alleles of the agent. The combinations of these genes influence the behaviour and decisions of the agent. The genes contain instructions that are interpreted as characteristics of the agents. Each agent follows its own strategy that is induced by the sequence of bits in its chromosome.

Table 1 describes the chromosome and the genes it contains, which encode the behaviour of the agent. The chromosome consists of 10 genes: Gene 1 is used to define the general character of the agent. Genes 2 to 5 define how the agent comes to trust decisions whereas gene 10 marks the actual trust decision. Genes 6 to 9 define, which characteristics of other agents are taken into account for decision making.

Each gene can have the value 1 or 0. Thus, there exist 2^{10} different types of Evolutionary Agents. Gene 1 defines the character of the agent. It decides whether the agent is an egoist (E) ($G_1 = 0$) or a cooperator (C) ($G_1 = 1$) and tries to do its job as well as possible. This means that an egoist will accept work units, if it trusts its partner, but will abort them with a high probability before they are finished. A cooperator will accept work units if the number of work units in its queue is not too high and it trusts its partner. Then it will try to process the work units and avoid aborting them. The genes 2 to 5 influence the decisions of the agent, if it will trust its partner. Those genes determine the signals which the agent has to pay attention to:

- Its own intentions (Gene 2)
- Reputation of the partner (Gene 3)
- Fitness of the partner (Gene 4)
- Workload of the partner (Gene 5)

The genes 6 to 9 determine how to interpret those signals. If the value of the signaling gene (2-5) is 0 then the corresponding gene (6-9) is ignored.

The following example (cf. Figure 1) will illustrate the signals and the corresponding behaviour of an agent A_i which decides how to interact with an agent A_j: We assume that the values of gene 2 and gene 4 are 1 and the values of gene 3 and gene 5 are 0. Gene 2 has the value 1, so its own intentions (Gene 1) will be included in the process of building trust. This depends on the value of gene 6. If gene 6 $=$ 0 the agent assumes that the partner is the opposite of the agent's gene 1. If gene 6 has the value 1, the agent assumes, that the partner's gene 1 has the same value as the agent's gene 1. The assumption that the partner is a cooperator will increase trust while the assumption that the partner is an egoist will decrease trust. Since gene 4=1, the agent pays attention to the partner's fitness. If gene 8=0 the agent will trust those which have a lower fitness than itself and distrust those with a higher fitness. If gene 8=1 the behaviour is inverted. The number of signals that recommend trust are normalized with the total number of signals that the agent pays attention to, so that it results in a value between 0 and 1. This value represents the probability that the agent will trust a partner. If an agent will pay attention to none of the four signals, then gene 10 decides if the agent will trust the partner. Based on total trust ($G_{10} = 1$) or total distrust ($G_{10} = 0$), the agent will accept all work units or reject all work units respectively.

Genes	**Alleles**	**Rules**
1	0	E (Egoist: aborts Work Units (WUs) with high probability).
	1	C (Cooperator: tries to process WUs as well as possible).
2	0	Don't involve your own intentions (G_1) into building trust.
	1	Involve your own intentions (G_1) into building trust, given (G_6).
3	0	Ignore the partner's reputation.
	1	Pay attention to the partner's reputation, given (G_7).
4	0	Ignore the fitness of the partner.
	1	Pay attention to the fitness of the partner, given (G_8).
5	0	Ignore the workload of the partner.
	1	Pay attention to the workload of the partner, given (G_9).
6	0	Assume that others are the opposite of your gene (G_1).
	1	Assume that others are the same as your gene (G_1).
7	0	Distrust those who have a relatively high reputation. Trust those who have a relatively low reputation.
	1	Trust those who have a relatively high reputation. Distrust those who have a relatively low reputation.
8	0	Distrust those who have a relatively high fitness. Trust those who have a relatively low fitness.
	1	Trust those who have a relatively high fitness. Distrust those who have a relatively low fitness.
9	0	Distrust those who have a relatively high workload. Trust those who have a relatively low workload.
	1	Trust those who have a relatively high workload. Distrust those who have a relatively low workload.
10	0	Distrust everybody (reject all WUs).
	1	Trust everybody (try to process all WUs).

Table 1: Chromosome Structure of the Evolutionary Agent

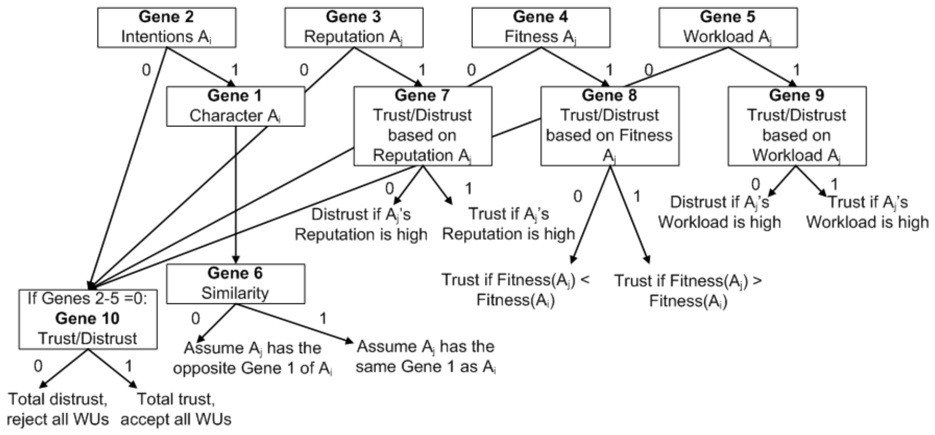

Gene 2
Intentions A_i
Gene 3
Reputation A_j
Gene 4
Fitness A_j
Gene 5
Workload A_j
0
1
Gene 1
Character A_i
Gene 7
Trust/Distrust based on Reputation A_j
Gene 8
Trust/Distrust based on Fitness A_j
Gene 9
Trust/Distrust based on Workload A_j
Distrust if A_j's Reputation is high
Trust if A_j's Reputation is high
Trust if Fitness(A_j) < Fitness(A_i)
Trust if Fitness(A_j) > Fitness(A_i)
Distrust if A_j's Workload is high
Trust if A_j's Workload is high
Gene 6
Similarity
Assume A_j has the opposite Gene 1 of A_i
Assume A_j has the same Gene 1 as A_i
If Genes 2-5 =0:
Gene 10
Trust/Distrust
Total distrust, reject all WUs
Total trust, accept all WUs

Fig. 1: The chromosome structure of Evolutionary Agent A_i decides how to treat agent A_j

3.1 Gene initialisation

The genes of each agent are set randomly at creation. For each gene there exists a parameter which has a value between 0 and 1. This parameter determines the probability that the corresponding gene is set to 1 when an Evolutionary Agent is created. Regarding the population of all Evolutionary Agents the value of the parameter corresponds to the expected number of agents which have the corresponding gene equal to 1. The standard value is 0.5, so that each gene of half of the Evolutionary Agents takes the value 1.

3.2 Evolution and spreading of the genes

To ensure the evolution of the genes and the corresponding trust strategies, a gene exchange between the agents can occur when they come into contact. In this process, the genetic instructions will be transferred with a fixed probability from the agents with a higher fitness to those who have a lower fitness. If two agents interact, the partner with the lower fitness replaces a random part of his chromosome structure with a part of the fitter partner's chromosome. In this procedure, each bit in a chromosome can be replaced independently of the others. Whether a bit of the agent with a lower fitness is replaced by the fitter agent's bit is determined by the recombination probability. In this case the recombination probability was 50%.

Furthermore, to increase heterogeneity, mutation occurs during the gene replacement. Thus, during the transfer of the genes from the fitter agent to the weaker agent random copying errors (mutations) can arise. The probability that a copying error during a gene replacement occurs is 1%. This value allows for sufficient heterogeneity without affecting the stability of evolution. Thus, Evolutionary Agents are able to leave local optima in their fitness landscape and have a higher probability to reach the global optimum.

3.3 Worker: Acceptance of Work Units

To decide whether a work unit is accepted or not the chromosome structure is analysed. If the agent pays attention to more than one signal of the partner, each bit is equally taken into account. Partners which send out mixed cooperation signals will be trusted with a corresponding probability. Let's assume that three of the signal genes are used, where two show trust in the partner and the third distrust. Then the agent will trust the partner and accept the work unit with a probability of $\frac{2}{3}$. In this paper, the signal genes are weighted equally.

3.4 Submitter: Distribution of Work Units

In our Grid agent model, a ranking of the suited worker agents is created to distribute the work units [2]. In this paper, this is done by calculating a score of reputation, fitness and workload whereby genes 3, 4 and 5 determine which of these characteristics are included and gene 7, 8 and 9 determine whether the total score will be increased or decreased. After creating the ranking, its own work unit is offered, in order of ranking, to the other agents until one of them accepts or the submitter has to process the work unit itself.

4 Experimental results

In the experiments presented in this paper, we investigated how the Evolutionary Agents behave with other types of agents and with each other. Subsection 4.1 introduces the system model and the parameters used in this evaluation. In Experiment 1 (Subsection 4.2), we analysed how Evolutionary Agents behave in a heterogeneous system of both Evolutionary and Adaptive Agents. Adaptive Agents have been the most successful agent type in former experiments [2]. Experiment 2 (Subsection 4.3) has been conducted in order to evaluate the behaviour in a homogeneous system of Evolutionary agents. The figures presented in this section show typical results of 10 runs conducted for each experiment.

4.1 System model

In the experiments, we observed 100 agents over a period of 100,000 ticks (time units). Each gene of a chromosome of an Evolutionary Agent is initialised with 1 with a probability of 50%. Additionally, each gene has a recombination probability of 50%, and the probability that mutation occurs during a recombination is 1%. The first experiment consisted of a population of 100 agents of which 50% were Evolutionary Agents and 50% Adaptive Agents. In the second experiment we generated a homogeneous population of 100 Evolutionary Agents. Other agent configurations have also been investigated, these two experiments have been chosen in order to cover both heterogeneous and homogeneous system configurations. The probability that an Evolutionary Agent with $gene_1 = 0$ aborts a work unit was set to 75% in both experiments.

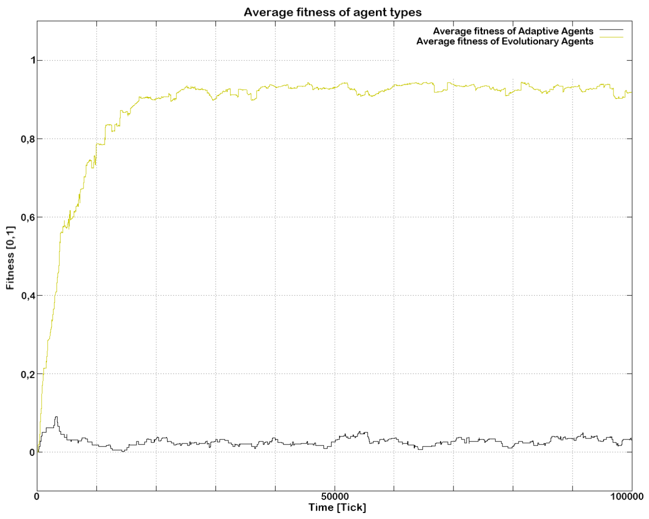
In both experiments, we measured the aggregated trust value $T_{i,j}$ of agent A_i in agent A_j as a function of the form

$$T_{i,j} = f(rep_j, exp_{i,j}), -1 \leq T_{i,j} \leq 1. \tag{1}$$

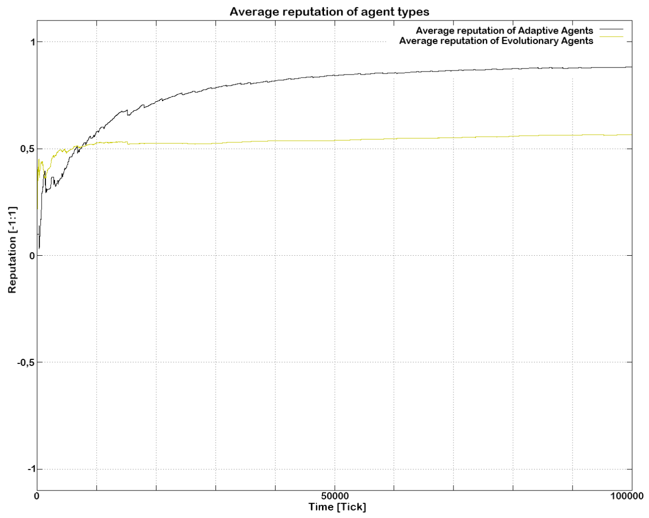
where rep_j is the reputation of the agent in the system and $exp_{i,j}$ an aggregation of the personal experiences A_i has had with A_j in the past. The function contains a weight: the fewer the agents' personal experiences, the higher the reputation weight. Nonetheless, the reputation is always taken into account to a certain degree in order to recognise changes in agent behaviour with more than just knowledge from own experience.

In both experiments, we measured the fitness of the agents. Our fitness function consists of the benefit the agent has from participating in the system as well as the effort he spent in order to reach this benefit:

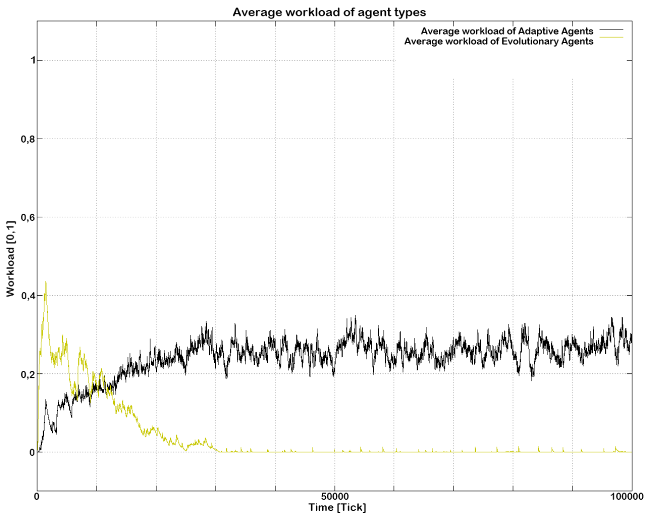
$$fitness = \alpha * benefit + (1 - \alpha) * (1 - effort) \tag{2}$$



(a) Average fitness of agents



(b) Average reputation of agents



(c) Average workload of agents

Fig. 2: Results of experiment 1

The fitness function is evaluated as soon as an agent has finished a job whose WUs have been distributed among the grid workers. The weight between benefit and effort is a factor α between 0 and 1 defined by the system designer. In these experiments, α was 0.8, which means that the benefit is valued much higher than the effort term. The agent fitness is between 0 and 1.

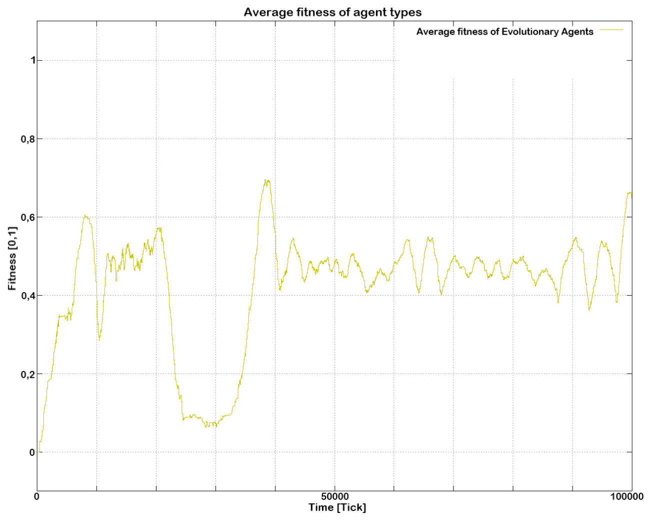
The $benefit$ is the time an agent has saved by distributing the job in the Grid rather than computing it on its own. In order to reach this benefit, the agents need an $effort$ in terms of gaining reputation by calculating WUs for other agents.

4.2 Experiment 1: Evolutionary agents vs. Adaptive Agents

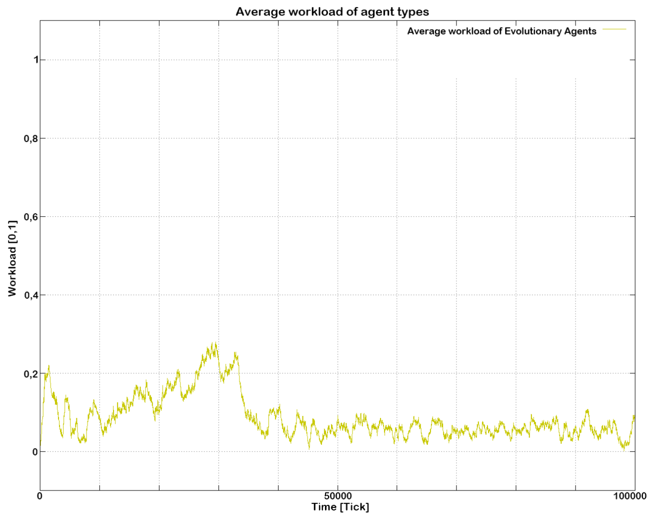
Figure 2a shows the average fitness of the Evolutionary Agents and the Adaptive Agents. The average fitness of the Evolutionary Agents is much higher than the fitness of the Adaptive Agents. The average reputation of the Adaptive Agents shown in Figure 2b is higher than the one of the Evolutionary Agents, but still the Evolutionary Agents have a good reputation greater than 0.5. Already after tick 30,000 a dominant chromosome structure has evolved: Gene 1, 4, 5, 6, 7, 8 and 9 have the value 1 and gene 2, 3 and 10 the value 0. Figure 2c shows that the workload of the Evolutionary Agents decreases and the workload of the Adaptive Agents increases which means that the Evolutionary Agents successfully distribute the work units to the Adaptive Agents. In other words, the Evolutionary Agents are able to exploit the Adaptive Agents. This behaviour can also be observed in systems with other agent types. Evolutionary agents are successful regardless of what the other agents in the system might be because their behaviour continuously adapts to the system configuration. The agents with the highest fitness are copied, thus, the most successful strategy for a given situation evolves and spreads over time. Therefore, in unknown system configurations, an evolutionary approach is worthwhile. This holds as long as there exist enough agents using this strategy. Our further experiments have shown that the enforcement of successful chromosomes needs about 25% of the system population being Evolutionary agents in order to be fast enough to successfully adapt to the environment.

4.3 Experiment 2: Homogeneous system of Evolutionary Agents

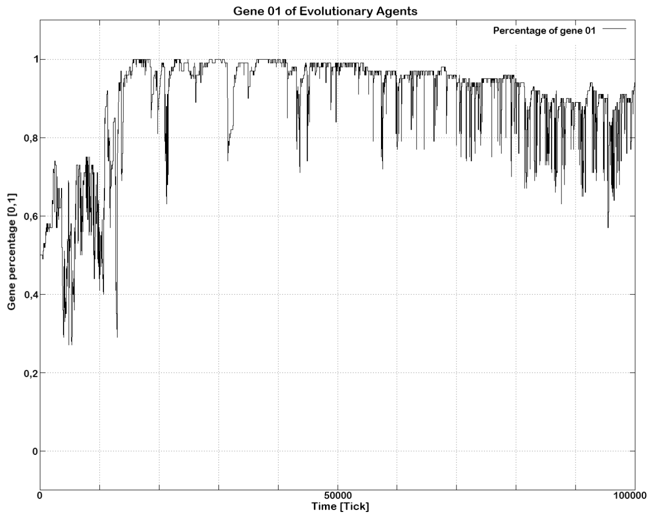
It can be seen from Figure 3c that value 1 for gene 1 wins from almost immediately after the start of the simulation. This shows that being cooperative is a more successful strategy than being egoistic. In Figures 3a it is particularly noticeable that at tick 20,000 the fitness strongly drops and rises again at tick 30,000. This matches with the fact that during the same period the value 0 of gene 4 in Figure 4a has established and so the largest part of the agents will not pay attention to the fitness of the partner in accepting work units. In Figure 3b it is noticeable that the increased workload in this period indicates that



(a) Average fitness of agents

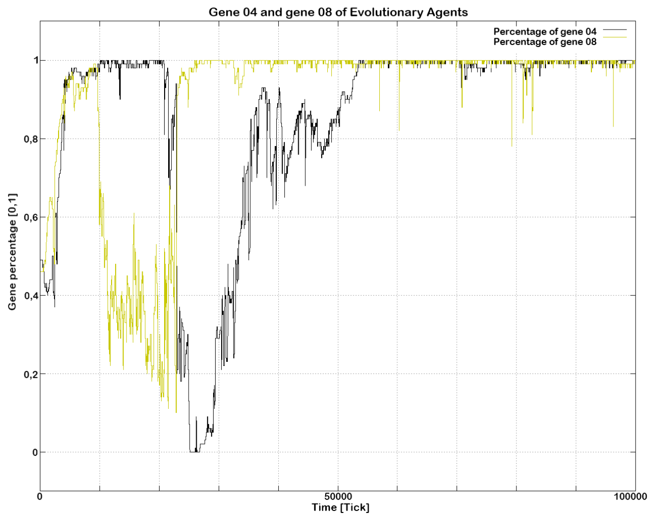


(b) Average workload of agents



(c) Gene 01 of Evolutionary Agents

Fig. 3: Results of experiment 2



(a) Gene 04 and gene 08 of Evolutionary Agents

Fig. 4: Results of experiment 2

due to the lack of trust the agents cannot distribute their work units any more and thus they have to process them on their own. As soon as the value 1 for gene 4 prevails, the fitness increases and the workload decreases again, because new trust is created between the agents. Thus, it is important to which of the chromosome signals the Evolutionary Agents pay attention: paying attention to the others' fitness is crucial to one agent's success. After gene 4 won through, the chromosome structure stabilizes and it is obvious that in this experiment it is not important for the Evolutionary Agents to pay attention to their own intentions based on gene 1 and to the reputation of the partners. Furthermore, it can be seen in Figure 4a that the amount of agents with a gene 8 develops to a high value. This means that agents trust agents with a fitness higher than their own, which also leads to a stable population of agents with a high fitness. Thus, agents pay attention to other agents' fitness (percentage of gene 4 near 1) and imitate agents with a high fitness (percentage of gene 8 near 1).

Overall, Evolutionary Agents are also successful in homogeneous systems, although the amplitude of the fitness is large as there are changes in genes while trying to adapt the chromosome structure to the self-referential fitness landscape in a continuously changing environment (cf. [3] (Chapter 2)).

5 Conclusion and Future Work

In this paper, the Evolutionary Agent as an approach to a self-optimising trust-adaptive agent has been introduced. The results of experiment 1 clearly show that, in a heterogeneous system, the Evolutionary Agent has a higher fitness than the Adaptive Agent, with a good reputation and low workload. Experiment 2 shows that Evolutionary agents are also able to interact in homogeneous systems as well. Therefore, the Evolutionary Agent approach promises to be well-suited as a run-time optimisation technique for Grid Computing agents. Further work will include the investigation of the suitability of Evolutionary Agents in large-scale systems as well as in systems with disturbances like Free Riders.

In the future of our project, we aim at investigating further learning techniques like Learning Classifier Systems, Neural Networks or Bayesian Networks in order to convey the optimisation more directly rather than relying on random mutation effects in order to make sure to find an optimal solution over time.

Furthermore, we will also investigate the system level of our Trusted Desktop Grid. The agents are able to form so-called Trusted Communities [2], i.e. agent organisations based on trust mechanisms. They are able to enhance the efficiency and robustness at both individual and system level.

6 Acknowledgements

This research is funded by the research unit "OC-Trust" (FOR 1085) of the German Research Foundation (DFG).

References

1. M. Amoretti. A framework for evolutionary peer-to-peer overlay schemes. In *Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, EvoWorkshops 2009, pages pp. 61–70. Springer-Verlag, 2009.
2. Yvonne Bernard, Lukas Klejnowski, Jörg Hähner, and Christian Müller-Schloer. Efficiency and robustness using trusted communities in a trusted desktop grid. In *Proceedings of the 2011 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop (SASOW)*. IEEE Computer Society Press, 2011.
3. E. Cakar. *Population-Based Runtime Optimisation in Static and Dynamic Environments*. PhD thesis, Leibniz Universität Hannover, 2011.
4. SungJin Choi, HongSoo Kim, EunJoung Byun, MaengSoon Baik, SungSuk Kim, ChanYeol Park, and ChongSun Hwang. Characterizing and classifying desktop grid. In *Cluster Computing and the Grid, 2007. CCGRID 2007. 7th IEEE International Symposium on*, pages 743 –748, 2007.
5. Ian Foster, Carl Kesselman, and Nicholas Jennings. Brain meets brawn: Why grid and agents need each other, 2004.
6. Michael W. Macy and John Skvoretz. The evolution of trust and cooperation between strangers: A computational model. Technical report, October 1998.
7. L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *Proc. 35th Annual Hawaii International Conference on HICSS System Sciences*, pages 2431–2439, January 7–10, 2002.
8. Christian Müller-Schloer and Hartmut Schmeck. Organic Computing - Quo Vadis? In Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors, *Organic Computing - A Paradigm Shift for Complex Systems*, chapter 6.2. Birkhäuser Verlag, 2011.