Parameter Control Using the Agent Based Patchwork Model

Thiemo Krink
Evalife project group, Department of Computer Science
Aarhus Universitet, Ny Munkegade, Bgd 540
DK-8000 Aarhus C, Denmark
krink@daimi.au.dk

Rasmus K. Ursem
Evalife project group, Department of Computer Science
Aarhus Universitet, Ny Munkegade, Bgd 540
DK-8000 Aarhus C, Denmark
ursem@daimi.au.dk

Abstract- The setting of parameters in Evolutionary Algorithms (EA) has crucial influence on their performance. Typically, the best choice depends on the optimization task. Some parameters yield better results when they are varied during the run. Recently, the so-called Terrain-Based Genetic Algorithm (TBGA) was introduced, which is a self-tuning version of the traditional Cellular Genetic Algorithm (CGA). In a TBGA, the individuals of the population are placed in a two-dimensional grid, where only neighbored individuals can mate with each other. The position of an individual in this grid is interpreted as its offspring's specific mutation rate and number of crossover points. This approach allows to apply GA parameters that are optimal for (i) the type of optimization task and (ii) the current state of the optimization process. However, only a few individuals can apply the optimal parameters simultaneously due to their fixed position in the grid lattice.

In this paper, we substituted the fixed spatial structure of CGAs with the agent-based Patchwork model. In this model individuals can move between neighbored grid cells, and the number of individuals per grid cell is variable but limited. With this design, several individuals were able to use beneficial parameters simultaneously and to follow optimal parameter settings over time. Our new approach achieved better results than our original Patchwork model and the TBGA.

1 Introduction

Several studies on Evolutionary Algorithms have been concerned with the choice of parameters for an optimal performance (e.g. [Grefenstette, 1986], [Lee and Takagi, 1993] and [Schaffer et al., 1989]). Unfortunately, this choice usually depends on the optimization task, which sets a limitation to the approach. Further, some parameters improve the performance significantly when they are varied over time such as the mutation rate or the variance of a Gaussian mutation operator. One idea to cope with this lack of generality is to use self-adaptive algorithms where the EA parameters become a part of the simulated evolutionary process (e.g. [Bäck, 1992]; for a survey on parameter control see [Eiben et al., 1999]). Though this idea attracted attention in recent years, it dates back to the very beginning of the field, when Rechenberg introduced evolutionary strategies [Rechenberg, 1973]. Here, the mutation rate becomes a part of the individual's genome.

A recent promising idea in the context of self-adaptation are the so called Terrain-Based Genetic Algorithm (TBGA) [Gordon et al., 1999], which is a self-tuning version of the traditional Cellular Genetic Algorithm (CGA) [Whitley, 1993] and [Bäck et al., 1997, part C6.4]. In TBGAs, the two-dimensional grid position of an individual is interpreted as its offspring's mutation rate and number of crossover points. Thus the individuals of a TBGA population apply the entire variety of different parameter combinations at each time step. The advantage of this approach is that GA parameters can be exploited that are optimal for the type of optimization task and the current state of the optimization process. However, since the individuals in the CGA are fixed at their grid positions, only a few of them are able to take advantage of this set-up simultaneously.

One way to tackle this problem is to use a multi-agent system where agents can move between grid cells and more than one agent can be at the same grid location. An useful approach in this context are Patchwork models, which we recently introduced as a new hybrid model for spatial population structure in Evolutionary Algorithms [Krink et al., 1999]. In the Patchwork model, individuals are considered as autonomous mobile agents that choose their actions based on "desires", such as to stay close to other agents with a high fitness. The main idea of this approach is to introduce spatial aspects to Evolutionary Algorithms, which are inspired by real world ecology. The result is a simplified spatial model that combines the concepts of Cellular Genetic Algorithms (CGA) [Whitley, 1993] with island models (e.g. [Krink and Vollrath, 1997] and [Bäck et al., 1997, part C6.3]). Like in CGAs, Patchwork models are based on a discrete spatial population structure organised in neighborhoods. However, in contrast to CGAs, Patchwork agents can move between separated patches that may contain several other agents, which corresponds to the concept of migration between subpopulations in island models.

In this paper, we substituted the fixed spatial structure in TBGAs with the Patchwork model. From this design, we expected that more agents could take advantage of beneficial parameters by moving to the grid location with the optimal settings. The paper is structured as follows: In the next section we will introduce the new version of the Patchwork model and a revised version of the motivation network model, which is used to control the agents. Afterwards, we will present some experiments where we compared the performance of the new Terrain-Based Patchwork model with the original Patchwork model and the TBGA and summarize the results in the last section.

2 The Patchwork Model

The Patchwork model was introduced in [Krink et al., 1999] and is a hybrid evolutionary algorithm implementing ideas from the cellular GA, multi-agent systems, island models and traditional genetic algorithms. A Patchwork model consists of a grid world and a number of interacting agents.

Agents contain a binary string representing a point in the search space of the objective function. In each time step, an agent's behavior is determined by its motivation network, which executes an action, such as moving, mating, and eating. The decision process is based on the input from a number of sensors and their "interpretation" by the motivation network. The sensors are capable of detecting various properties, such as best fitness, population density, and own fitness. We defined two kind of sensors: 1) External sensors returning a value based on the state of the patch it is scanning, and 2) internal sensors reflecting the internal state of the agent. Each sensor has an output value between 0 and 1.

For each possible action the motivation network has a set of functions that map the sensor values to a motivation for performing that action. For instance, a setting with two actions and four sensors implies that the motivation network will have eight mapping functions. These functions are encoded in the agent's genome so it can evolve behavior patterns like "follow agents with good fitness, but avoid overcrowded and empty patches". Each sensor is represented by six values, a weight and five points defining the mapping function. See figure 1 for an example. The sensor values are combined to a motivation value M_i as follows:

$$M_i = \sum_{j=1}^{\#sensors} w_{ij} \cdot map_{ij}(s_j)$$

where w_{ij} is a weight between 0 and 1 for sensor j in action i, map_{ij} is the mapping function and s_j is the sensor value.

Figure 1 illustrates an example of an agent's mapping functions for moving when the model supports sensors for best fitness and patch population density. In this case, the agent will be motivated to move to patches with density near 0.5 and high fitness.

The action of an agent is determined by calculating the motivation for each available action on each of the patches in the agent's neighborhood. If the agent can choose between two actions, e.g. moving and mating, and the neighborhood range is 2 the motivation will be calculated for 50 action-patch pairs. (A range of 2 implies 25 neighborhood patches, whereas a range equal to 3 corresponds to 49 patches). The motivation network is illustrated in figure 2. The "action selector" is responsible for selecting the agent's next action

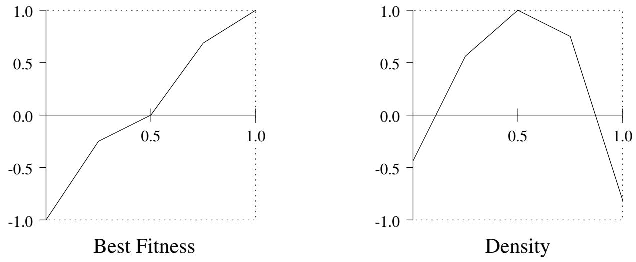


Figure 1: Example with sensor mapping functions.

based on the motivation values. We used a simple max selector, which selected the action with the highest motivation. In case that multiple action-patch pairs evaluate to the same high motivation, a random action-patch is selected and the action with the corresponding patch is performed.

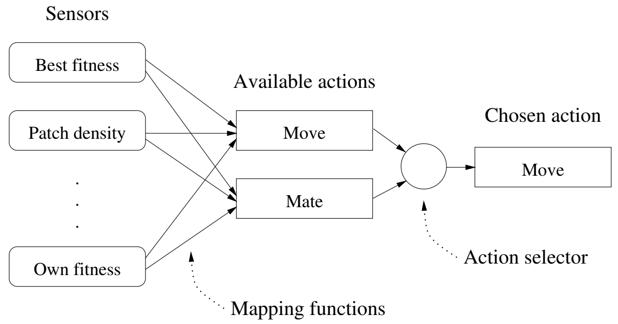


Figure 2: Motivation network.

In the study presented in this paper, we defined the Patchwork world as a two dimensional grid with connected borders (a torus). The patches in this grid world had a carrying capacity determining the maximal number of agents allowed on a single patch. This capacity was set to 4, which forced the agents to inhabit several grid cells of the world. An agent was allowed to interact with other agents within its range. Figure 3 illustrates a 10×10 world and an agent with a neighborhood range of 2 patches.

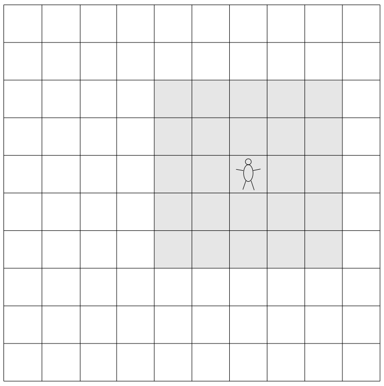


Figure 3: 10×10 world and an agent with a range of 2.

For the control of the agents, we tested the following sensors:

1. Relative best fitness in current iteration.
   The best and worst fitness are determined for the whole population. The fitness of the best agent on the scanned patch is scaled to a value between 0 and 1 according to the best and worst fitness in the population.

2. Relative best fitness since iteration 0.
   The best and worst fitness are stored over all iterations. The fitness of the best agent on the scanned patch is scaled according to these values.

3. Patch population density.
   This sensor indicates the crowdedness of a patch (0 means no agents; 1 means that the number of agents on the patch is equal to the carrying capacity).

4. Own fitness.
   The agent's own fitness is scaled according to the best and worst fitness since iteration 0.

Our preliminary experiments showed that simple agents were able to find good parameters much faster than complex agents with many sensors. Based on these experiments, we decided to use only the sensor for the relative best fitness within the current iteration. The motivation network implemented the move action, whereas recombination was performed separately in each timestep. Since our objective was to track parameters leading to good solutions we predefined our agents to follow agents with good fitness.

When an agent decided to move to a new patch it tried to go there as directly as possible. If the direct way was blocked because of overcrowding, it tried to find an alternative route. As illustrated in figure 4 there are two kind of moves in the grid world, axis-parallel moves and diagonal moves. In this example, the agent will first try to move in the direction of the thick arrow. Otherwise, if that patch is full, it will try to go in one (randomly picked) direction of the medium thick arrows. If both are blocked it will try the two directions indicated by the thin arrows. If all of the five directions are blocked the agent will stay where it is. This moving algorithm lead to an agent behavior, where agents formed circular clusters, because they tried to get as close as possible to the desired destination.

In each timestep all agents with a potential mate in their neighborhood were mating with a random agent on their own patch or, with a lower probability, with an agent from a neighbored patch. After recombination the standard bitflip mutation operator was applied to the offspring. The first parent was replaced by the offspring in case that the offspring had a better fitness.

2.1 Parameter Space Encoding

The location of a parent agent in the patch world determined the number of crossover points and mutation rate of the offspring produced by the recombination operator. Figure 5 illustrates a 10×10 world with one agent. The offspring of the agent will be generated using 3 crossover points and a mutation rate of $0.6/genomelength$.

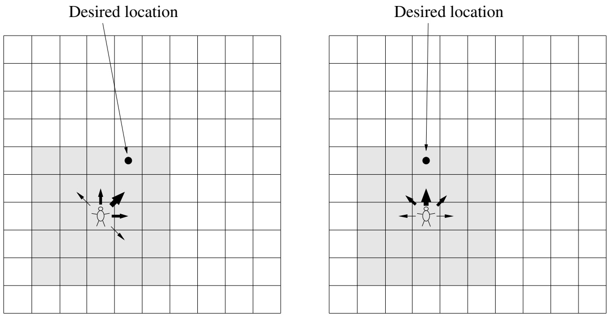


Figure 4: Example of the two kind of moving.

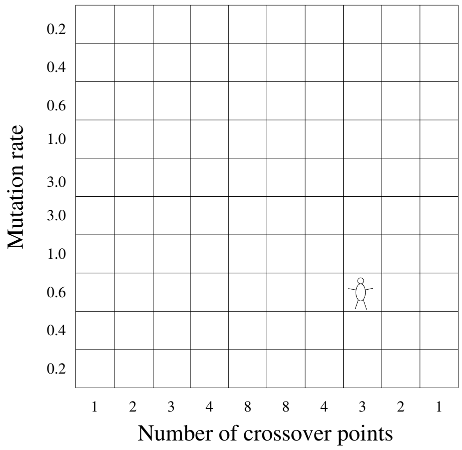


Figure 5: Example of 10×10 gridworld with one agent.

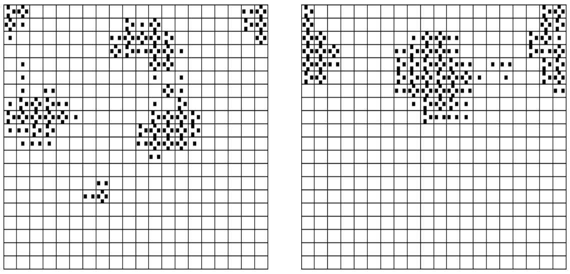
In the experiments we used a 20×20 world with the following crossover points (0, 1, 2, 3, 4, 5, 6, 7, 11, 17) distributed over the first 10 columns and reversed for the last 10. The mutation rates were (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 1.31, 2.44) divided by the length of the genome and distributed over the rows like the crossover points.

3 Experiments

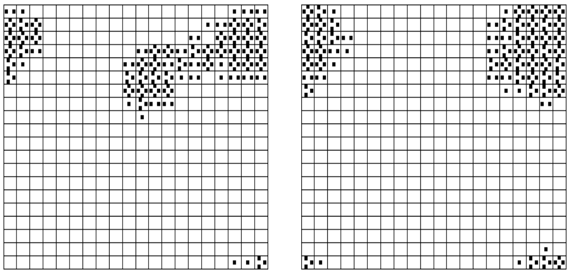
The primary objective of the experiments was to test whether the agents that carried the best solutions would flock in a certain area of the parameter space. Further we tested whether the Patchwork model would lead to better results compared to the TBGA. The preliminary runs indicated a special flock-moving pattern where the agents appeared to follow the same path through the parameter space during the runs. One particulary interesting observation in these experiments was that some parameter settings appeared to be beneficial at certain periods of the run.

3.1 Example of flocking behavior

Figure 6 illustrates a typical run with 200 agents that move every generation. The algorithm is initialized with a maximum of one agent per patch, i.e. the start position of the agents covered many different parameter settings. After a couple of generations the agents formed small clusters, which then merged to larger clusters and finally into one single cluster. In generation 10 two smaller clusters merged to one in generation 25.



(a) Generation 10. (b) Generation 25.



(c) Generation 40. (d) Generation 60.

Figure 6: Example of the flocking behavior.

Figure 6(c) shows the situation where the center cluster began to merge to the cluster located in the corners[1], which was completed in generation 60.

3.2 Test functions

The following five test functions were used in the experiments: De Jong F2+F4, Rastrigin F1 (20D), Schaffer F6 and Krink F1. They were selected to allow a comparison between the new Terrain-Based Patchwork model previous work on the Patchwork model and comparison between the Patchwork model and the TBGA found in [Gordon et al., 1999]. The test functions are defined as follows:

De Jong F2:

$$f(x,y) = 100(x^2 - y)^2 + (1-x)^2$$

where

$$-2.048 \leq x \leq 2.048 \text{ and } -2.048 \leq y \leq 2.048$$

De Jong F4:

$$f(\overline{x}) = \sum_{i=1}^{30} x_i^4$$

where

$$-1.28 \leq x_i \leq 1.28$$

Rastrigin F1 (20 dimensions):

$$f(\overline{x}) = 200 + \sum_{i=1}^{20} x_i^2 - 10 \cdot cos(2\pi x_i)$$

where

$$-5.12 \leq x_i \leq 5.12$$

Schaffer F6:

$$f(x,y) = 0.5 + \frac{sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{\left(1 + 0.001(x^2+y^2)\right)^2}$$

where

$$-100 \leq x \leq 100 \text{ and } -100 \leq y \leq 100$$

Krink F1:

$$\begin{aligned} f(x,y) &= 50 - \|x - 50\| + 40 sin\left(\frac{5\pi \cdot x}{18}\right) + \\ &\quad 50 - \|y - 50\| + 40 sin\left(\frac{5\pi \cdot y}{18}\right) \end{aligned}$$

where

$$0 \leq x \leq 100 \text{ and } 0 \leq y \leq 100$$

The Krink F1 has many local optima and one global optimum at approximately (52.167,52.167). The function is plotted in figure 7.

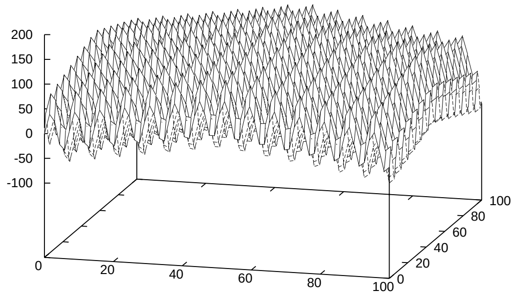


Figure 7: The Krink F1

[1]The world was defined as a torus so even though it looks like as if the population is scattered over four corner clusters it is in fact a single cluster.

3.3 Settings for the algorithm

The genome representing solutions were encoded as a binary string with 20 bits per variable. The population size was either 200 or 400 agents. Since the grid was 20×20 the TBGA had 400 agents. The carrying capacity of a patch was four agents. The probability for picking a mate on a neighbored patch was 50%. The agents had a sensor range of four cells and were allowed to move every five generations. As described in section 2, the motivation network was predefined so that agents were attracted to the fittest agent in their neighborhood.

3.4 Data sampling

For the comparison between the Patchwork model and the TBGA we used the average best fitness of 30 runs. We recorded the development of the mutation rate and the number of crossover points through a run by calculating the average parameter values corresponding to the position of the best agents over the last 10 generations.

3.5 Results

First we compared the combination of the TBGA and the Patchwork model (from now on called TB Patchwork model) with the original Patchwork model. In this experiment we found that the TB Patchwork model outperformed the original Patchwork model for the testfunctions Krink F1 and Schaffer F6 (fig. 8 and 9). Even with 200 agents the TB Patchwork model performed equally well or better than the original Patchwork model with about 500 agents.

In our second experiment we compared the TB Patchwork model with the TBGA. Here we found that the former produced better results for Krink F1 and De Jong F2 and comparable results for Schaffer F6, De Jong F4 and Rastrigin F1. Also in this comparison the TB Patchwork model with 200 agents had a better performance than the TBGA with 400 individuals for the Krink F1 and the De Jong F2.

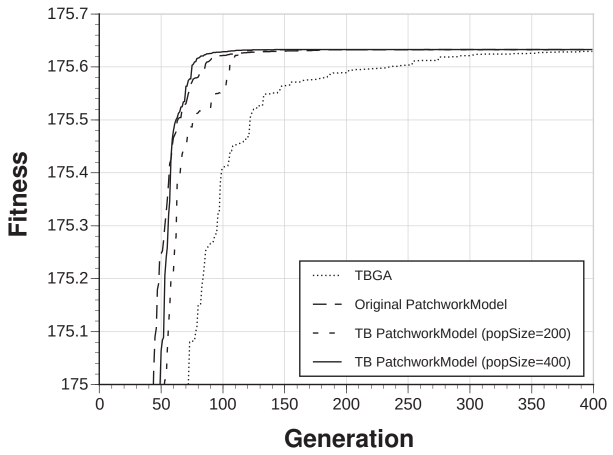


Figure 8: Average best fitness for the Krink F1 (30 runs).

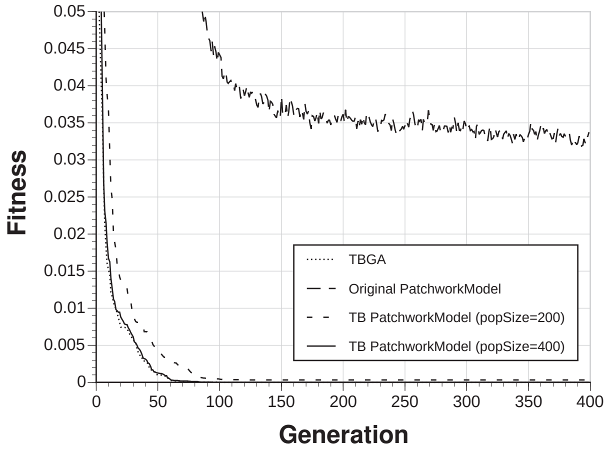


Figure 9: Average best fitness for the Schaffer F6 (30 runs).

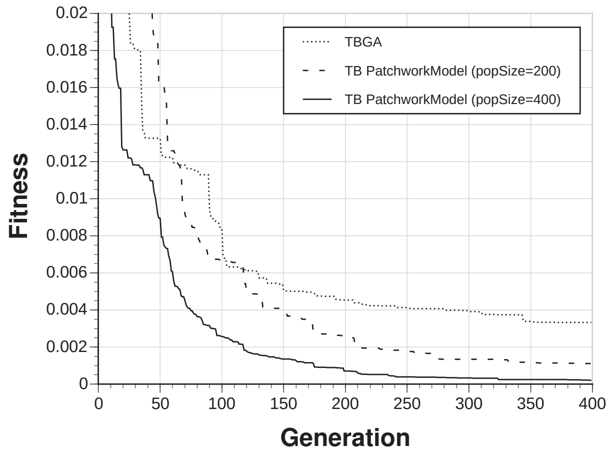


Figure 10: Average best fitness for the De Jong F2 (30 runs).

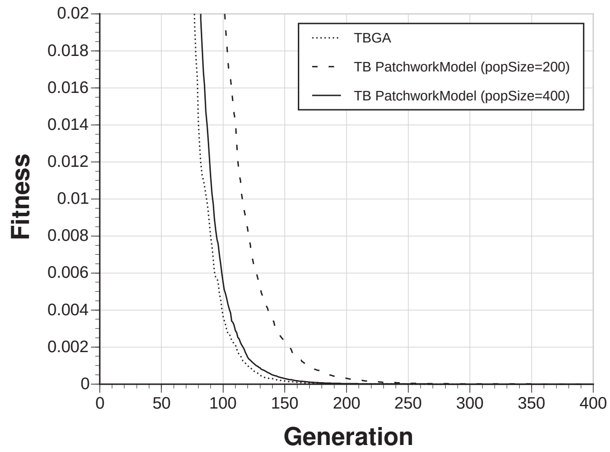


Figure 11: Average best fitness for the De Jong F4 (30 runs).

Regarding the change of the GA parameter values over time we got very distinct results for the different test func-

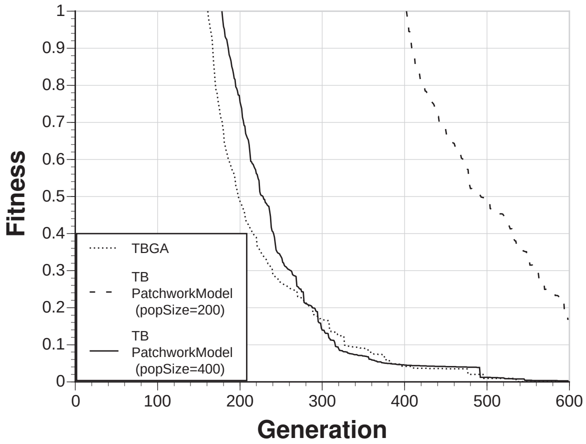


Figure 12: Average best fitness for the Rastrigin F1 (30 runs).

tions. In case of Krink F1 and De Jong F2 (fig. 13) the mutation rate dropped from an initial value of about 0.72 to a value below 0.1. The slow and steady increase of the mutation rate in case of Krink F1 was probably due to unspecific drift, since the algorithm already discovered the optimum after about 100 generations, and thus the optimal GA parameter settings were unable to improve the fitness results any further.

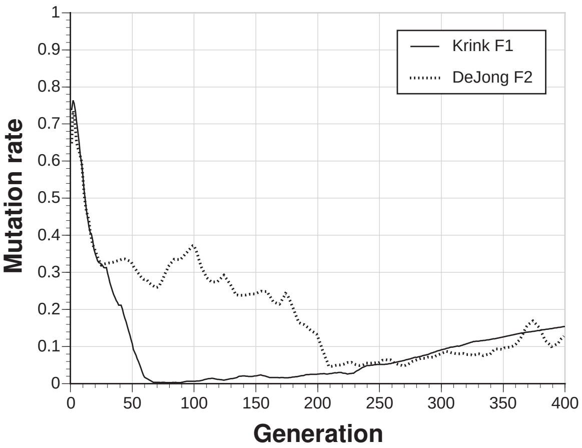


Figure 13: Timevarying mutation rate for Krink F1 and De Jong F2.

In case of the Schaffer F6, the Rastrigin F1, and the De Jong F4 (fig. 14) the mutation rate was much higher (between 0.5 and 0.9) throughout the run and did not settle on a value for the latter two functions.

The number of crossover points decreased for Krink F1 (fig. 15) and De Jong F4 from about 6 to 4, whereas there was an increase from about 5 to 7 for De Jong F2 and Rastrigin F1 (fig. 16) and a rather unsteady pattern for Schaffer F6 between 4 and 6.

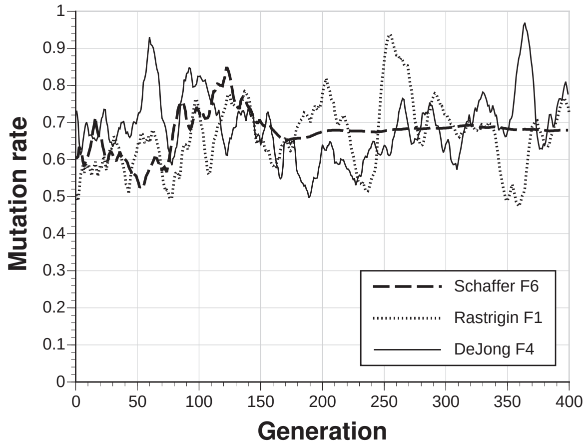


Figure 14: Timevarying mutation rate for Schaffer F6, Rastrigin F1 and De Jong F4.

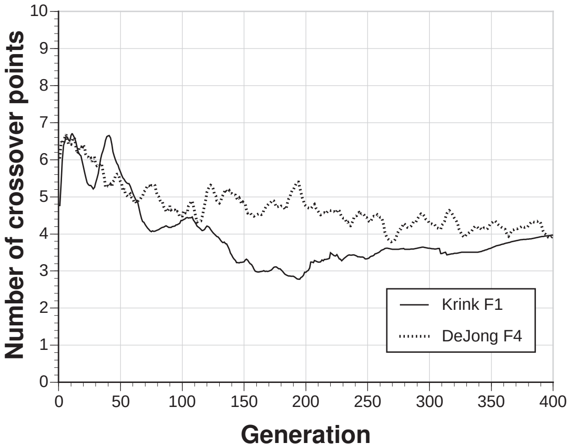


Figure 15: Timevarying number of crossover points for Krink F1 and De Jong F2.

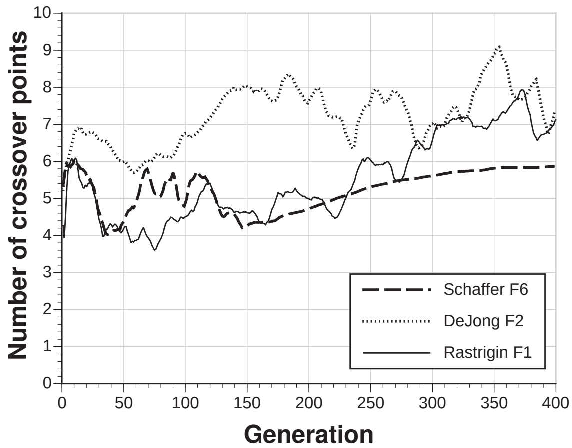


Figure 16: Timevarying number of crossover points for Schaffer F6, Rastrigin F1 and De Jong F4.

4 Conclusions

The Terrain-Based GA (TBGA) is known to perform better than traditional CGAs and much better than standard GAs for several test functions including De Jong's F2 and Rastrigin F1 [Gordon et al., 1999]. The results of our experiments showed that our Terrain-Based Patchwork model could further improve the performance of the original TBGA model in two test cases (Krink F1 and De Jong F2) and the original Patchwork model (on Krink F1 and Schaffer F6) even with half the number of individuals. In case of the Schaffer F6, the De Jong F4, and the Rastrigin F1 function the Terrain-Based Patchwork model and the TBGA performed equally well.

Compared to the TBGA our approach introduces four additional parameters concerning the sensor range, the carrying capacity per cell, the move interval, and the probability of mating with an agent of a neighbored cell. However, the optimal choice of these parameters appeared to be independent of the test problems.

During the runs, the Patchwork agents formed flocks around the most beneficial parameter settings by following the agents with the highest fitness. For instance, during the experiments with the Krink F1 and De Jong F2 function, the agents consistently moved from areas with a rather high mutation rate of about 0.7 to areas with a low mutation rate below 0.1. Interestingly, the preferred mutation rate increased slightly for both functions towards the end of the run. We interpret this result as a 'drift effect' after the population already discovered the global optimum of the test function.

In connection with the changing mutation rate we observed two clear tendencies, either (i) an annealing effect, where the mutation rate decreased to a low value or (ii) a more or less constant value. Interestingly, these are the two heuristics used to control the parameters in most evolutionary algorithms. Furthermore, we observed that the the annealing rates for the Krink F1 and De Jong F2 were quite different, which also supports the general hypothesis that optimal parameters are problem dependent.

Our results show that agent-based hybrid systems can be a powerful alternative for traditional spatial models such as island models and CGAs.

One extension of this work would be to investigate the terrain-based self-adaptation of other EA parameters. Furthermore, more than two parameters could be controlled simultaneously by increasing the dimensionality of the grid world. Finally, we would expect that the terrain-based approach turns out to be robust in context of dynamic environments. Here, the change of the fitness landscape requires a rapid adjustment of the EA parameter settings during the run [Ursem, 2000].

References

[Bäck, 1992] Bäck, T. (1992). Self-adaptation in genetic algorithms. In F. J. Varela, P. B., editor, *Proceedings of 1st European Conference on Artificial Life*, pages 263–271.

[Bäck et al., 1997] Bäck, T., Fogel, D. B., Michalewicz, Z., et al., editors (1997). *Handbook on Evolutionary Computation*. IOP Publishing Ltd and Oxford University Press.

[Eiben et al., 1999] Eiben, A. E., Hintering, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141.

[Gordon et al., 1999] Gordon, V. S., Pirie, R., Wachter, A., and Sharp, S. (1999). Terrain-based genetic algorithm (TBGA): Modeling parameter space as terrain. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 229–235.

[Grefenstette, 1986] Grefenstette, J. J. (1986). Optimisation of control parameters for genetic algorithms. *IEE Trans SMC*, 16(1):122–128.

[Krink et al., 1999] Krink, T., Mayoh, B. H., and Michalewicz, Z. (1999). A PATCHWORK model for evolutionary algorithms with structured and variable size populations. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1321–1328.

[Krink and Vollrath, 1997] Krink, T. and Vollrath, F. (1997). Analysing spider web-building behaviour with rule-based simulations and genetic algorithms. *Journal of Theoretical Biology*, 185:321–331.

[Lee and Takagi, 1993] Lee, M. and Takagi, H. (1993). Dynamic control of genetic algorithms using fuzzy logic techniques. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*.

[Rechenberg, 1973] Rechenberg, I. (1973). *Evolutionsstrategie*, volume 15 of *problemata*. Friedrich Frommann Verlag (Günther Holzboog KG).

[Schaffer et al., 1989] Schaffer, J. D., Caruana, R. A., Eshelman, L., and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*.

[Ursem, 2000] Ursem, R. K. (2000). Multinational gas: Multimodal optimization techniques in dynamic evironments. In *Proceedings of the Second Genetic and Evolutionary Computation Conference*.

[Whitley, 1993] Whitley, D. (1993). Cellular genetic algorithms. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, page 658.