Intelligent Exploration for Genetic Algorithms

Using Self-Organizing Maps in Evolutionary Computation

Heni Ben Amor
Universität Koblenz-Landau
Universitätsstraße 1
56070 Koblenz, Germany
amor@uni-koblenz.de

Achim Rettinger
Universität Koblenz-Landau
Universitätsstraße 1
56070 Koblenz, Germany
achim@uni-koblenz.de

ABSTRACT

Exploration vs. exploitation is a well known issue in Evolutionary Algorithms. Accordingly, an unbalanced search can lead to premature convergence. GASOM, a novel Genetic Algorithm, addresses this problem by intelligent exploration techniques. The approach uses Self-Organizing Maps to mine data from the evolution process. The information obtained is successfully utilized to enhance the search strategy and confront genetic drift. This way, local optima are avoided and exploratory power is maintained. The evaluation of GASOM on well known problems shows that it effectively prevents premature convergence and seeks the global optimum. Particularly on deceptive and misleading functions it showed outstanding performance. Additionally, representing the search history by the Self-Organizing Map provides a visually pleasing insight into the state and course of evolution.

Categories and Subject Descriptors

I.1.2.8 [**Computing Methodologies**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

Genetic Algorithm, Self-Organizing Map, Exploration vs. Exploitation, Diversity, Premature Convergence, Genetic Drift

1. INTRODUCTION

Techniques from the field of Evolutionary Computation, in this case Genetic Algorithms (GA), have been proven to be well suited for finding global optima in complex search spaces. Using a population of individuals evolving over numerous generations as a metaphor the search is iteratively guided by the fitness of the current parent generation. During an optimization run thousands of individuals, each one representing a possible solution, are generated, evaluated and by chance recombined to produce offspring. Information from previous generations is only implicitly and partially preserved in the current genome.

This bears the risk of a regeneration of individuals that have already been seen in the search process. Even more problematic is the fact that the search can be negatively affected by genetic drift. As a consequence, big parts of the search space, potentially containing the global optimum, will never be explored.

In this work we try to show that there is an efficient way not only to monitor the whole evolution process but also to extract valuable data from it (see section 4) which is used to guide the search process (see section 5). This task is achieved by adaptive operators utilizing data, mined by a Self-Organizing Map (SOM), from individuals of previous generations. The evaluation of our approach proves that GASOM is a well suited tool for addressing the issue of premature convergence in GAs (see section 6). Finally we point out how other problems can be solved by GASOM (see section 7).

GECCO'05, June 25–29, 2005, Washington, DC, USA.

2. BACKGROUND

2.1 Premature Convergence and the Loss of Diversity

A critical problem when dealing with Evolutionary Algorithms (EA) is the phenomenon of premature convergence. According to [4], premature convergence occurs when the population of a GA reaches a suboptimal state where genetic operators can no longer produce offspring which outperforms their parents. In that case, the search process will likely be trapped in a region containing a non-global optimum. A simple yet popular explanation for the occurence of this phenomenon is the loss of diversity. In that context diversity refers to the (genetic) variation of the population members. Previous work on tackling premature convergence was mostly centered on diversification techniques. Various enhanced strategies for diversity maintenance have been proposed which target the different stages of the evolution process.

Selection

Some techniques try to prevent selection from being too much biased towards high fitness individuals. Usually this is

done by pre-processing the fitness values prior to the selection procedure. Rank scaling [5], for instance, ranks the individuals according to their raw objective value. This avoids the possibility of a small number of highly fit individuals dominating the reproduction process. On the other hand, so called sharing methods [14] force an individual to share its fitness with other population members occupying the same niche.

Mating
Another strategy to maintain high diversity is to apply restriction and encouragement to the selection of the mating partners. In [1] incest is prevented by prohibiting crossover of genetically similar individuals. More specifically, mating partners with a Hamming distance below a given threshold were not authorized for crossover. Seduction [13] tries to mimic the mating behavior of sexually reproducing organisms. Here, the mating decision is based upon an attraction value computed between the partners.

Replacement
In steady-state GAs the replacement strategy determines which elements will lose their place in the population when new chromosomes are inserted. A well designed replacement condition can keep promising genetic material from getting lost. In deterministic crowding [9], each child replaces the most similar parent if it has higher fitness. As parents and children typically are very similar, unfair competition between individuals occupying different niches is avoided. Recently, a hybrid replacement scheme was proposed by Lozano et al[8]. They try to replace chromosomes which perform poorly with respect to both fitness and contribution to diversity.

Although the extensions described above have been shown to partially improve the efficiency of the search, they make simplifying assumptions by attacking rather the symptoms of premature convergence than the real cause [6].
First of all, high diversity does not imply better GA performance. This is tightly coupled to the question of exploration vs. exploitation. Enforcing diversity in the early phases of evolution ensures a broad exploration of the search space. However, in phases when high exploitation is needed, such promotion could be counterproductive or even destructive. Finding a modifiable balance between exploration and exploitation is the key to this issue.
Furthermore, once diversity is lost exploratory power cannot be solely regained by selection, mating or replacement of current individuals. Additionally, high mutation rates capable of increasing the diversity again are known to be mainly destructive. Hence there is need for an intelligent operator, which is able to reintroduce fresh and potential genetic material into the gene pool, even in highly converged populations.
Finally, previous approaches tried to ensure high genetic variability for each generation individually. Considering only the diversity of the current population, important regions of the solution space are possibly neglected, while others are revisited numerous times. As will be shown in this paper, the search can be controlled and guided more efficiently if diversity is considered throughout the whole evolution process. We refer to this as *novelty*.

2.2 The Self-Organizing Map

The SOM [7] is a class of artificial neural networks which has proven to be a valuable tool in analysis and visualization of high-dimensional data. Based on unsupervised learning the SOM performs a non-linear mapping from a high-dimensional input space onto a normally two-dimensional grid.

More precisely speaking, the SOM consists of a set U of units or neurons arranged on a regular grid. Each unit $i \in U$ is assigned a prototype vector $m_i = [m_{i1}, m_{i2}, ..., m_{in}]$ where n is the number of dimensions of the input space. The neurons are connected to adjacent neurons by a neighborhood relation which constitutes the topology of the map. Usually, a rectangular or hexagonal topology is used. Throughout this paper the rectangular topology will be used as this facilitates the visualization of each unit's prototype vector.

The SOM has successfully been applied in a wide range of research areas covering data mining, pattern recognition and most interestingly in the analysis and control of complex systems.

3. THE GASOM APPROACH

GASOM stands for "Genetic Algorithm using Self-Organizing Maps." This novel approach aims at avoiding the problems of GAs discussed in section 2.1 through information obtained from the evolution process by a SOM. These are its main features:

1. An explicit representation of the search history
2. A fitness evaluation promoting novelty
3. A reseeding operator preserving exploratory power
4. A control mechanism balancing exploration and exploitation

GASOM is based on a standard steady-state GA. In the next two sections we will elaborate on its improvements in detail. First, we will show in section 4 how SOMs can be used to monitor, document and analyze the behavior of EAs during an optimization run. The representation of the search history developed in this section, will help to gain insight into phenomena occurring during evolution and extract useful information from it. Section 5 comments on how the extracted information can be used to guide the search process efficiently.

4. MINING THE EVOLUTIONARY SEARCH

Although working in an algorithmically simple manner, EAs can produce vast amounts of data during an optimization run. In each of the numerous generations a large number of chromosomes is generated and evaluated. In traditional GAs only the current population is stored. But with the help of data from the previous generations you could gain valuable insight into the way EAs work and the problems they encounter. Possibly, we could draw important conclusions on how to improve their efficiency and guide the search online. Thus, there is need for an intelligent and efficient way of mining and storing the data during computation in real time. Necessary requirements are for example, processing incoming data such that a new chromosome should not lead to an entire recalculation of previously

computed knowledge. In other words, the knowledge acquisition should be incremental. Furthermore, the employed technique should be able to analyze data of arbitrarily high dimensions and independent from the number of data entities processed. Additionally, the computational overhead caused by its application should be scalable according to the needs. Finally, to facilitate human introspection and qualitative analysis, the applied technique should support some kind of graphical output.

The SOM algorithm introduced in 2.2 meets the discussed requirements. It projects the data samples onto a two-dimensional lattice. In contrast to many other multidimensional scaling methods, this projection does not have to be repeated when new data points are evaluated. It can be stored as a table and has low memory demands. The tabular representation facilitates the quantitative analysis of the recorded data. It can also easily be turned into various visualization forms including bitmaps, histograms or trajectories. In [12] SOM's have already been used to visulaize various aspects of evolutionary search.

Before applying, the SOM has to be appropriately trained in order to represent the envisioned data set. In this case the focus is on the genotype or solution space. Training the SOM with a large number of points from this space yields a two-dimensional projection of it.

4.1 Training the SOM

In this section how to train a SOM and how to use it to analyze the population is discussed. The training is carried out only once before the start of the GA. As long as the chromosome length and the representation form (binary vs. real coded) does not change, a trained SOM can be stored and reused. In the training phase a large number of different individuals from the genotype space are shown to the map. In our experiments a set of 100.000 different individuals and a map with 10x10 neurons were used.[1] The number n of weights in the prototype vectors equals the number of genes in a chromosome. The training comprised the following steps:

1. **Initialization**: Choose random values for the weights $m_{i1}, m_{i2}, ..., m_{in}$ of the prototype vectors. As the only restriction, the weight vectors m_i have to be different.

2. **Stimulus and Response**: A sample chromosome x is randomly chosen from the genotype space. In this step it is crucial that the random selection of chromosomes obeys a uniform distribution. If not, the learning could be biased towards a particular region of the search space. Distances to all prototype vectors are then computed via some distance measure (usually Euclidean distance). The winning neuron b, also called best-matching unit (BMU), is the map unit with prototype closest to x.

$$\|x - m_b\| = \min_i\{\|x - m_i\|\} \quad (1)$$

3. **Adaptation**: The prototype vectors are then updated according to the following update rule:

$$m_i(t+1) = m_i(t) + \alpha(t)h_{bi}(t)[x - m_i(t)] \quad (2)$$

where t is the current iteration, $\alpha(t)$ is the learning rate at iteration t and $h_{bi}(t)$ is a neighborhood kernel centered at the winning unit:

$$h_{bi}(t) = exp \quad -\frac{\|r - r'\|^2}{2\sigma^2(t)} \quad (3)$$

where r and r' denote the positions of the BMU and the neuron i on the SOM grid. Both $\alpha(t)$ and $\sigma(t)$ are monotonically decreasing functions of time.

Step 2 and 3 are repeated until the maximum number of iterations is reached. Once training is finished, each unit stands for a particular region of the solution space. It represents all chromosomes to which it is closest.

The learning process, as described above, is of stochastic nature. It also expects various parameters like the learning rate to be set by the user. Consequently, the difference in accuracy between two trained maps can vary heavily. If we would apply a map of mediocre accuracy to analyze a given data set, we would probably draw wrong conclusions. Hence we need to ensure that a map is of reliable quality before we can use it for the envisioned tasks.

In this work a simple test procedure is employed to test the quality of a SOM.We randomly generate a set of new chromosomes and project them onto the map. The projection is done by finding the appropriate BMU for each of these chromosomes, as defined by equation 1. We keep track how many times each neuron was activated as the BMU. Let $E(i)$ be a function, that returns the number of times a neuron i was activated. Ideally, the activation frequency of all neurons is equal:

$$E(i) = E(j) \quad \forall i, j \in U \quad (4)$$

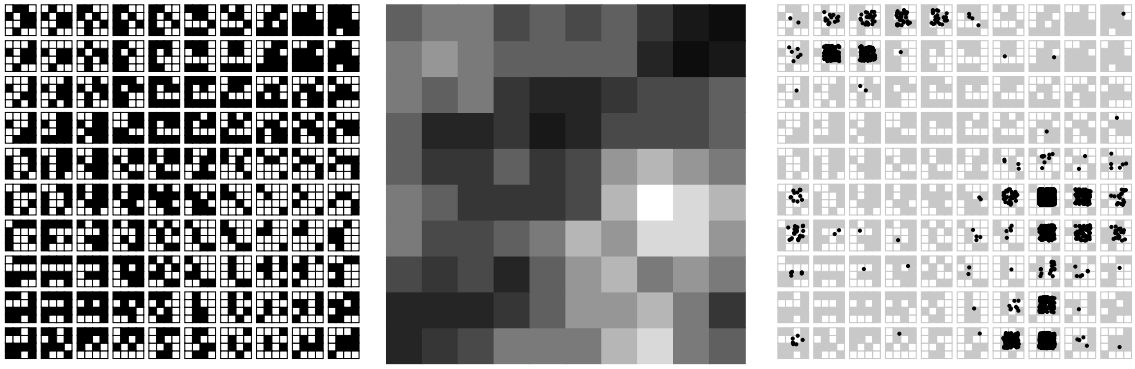
Thus, in order to ensure reliable analysis, we take a SOM which fits requirement 4, best. If it does not, we assume that the learning was biased.

4.2 Mapping the Population

Now, we can employ the trained SOM to classify incoming data/chromosomes. For this we assign each evaluated individual to the nearest map unit, as already done before in the optimization step. To store the resulting information, we will use two frequency tables. The first table, which we will call the *population distribution table*, holds the activation frequencies with respect to only those individuals that are currently in the population. In the following the activation frequency of a neuron i in the population distribution table, will be denoted by $E_p(i)$. The second table, called the *search history table*, stores the activation frequencies with respect to all individuals evaluated during evolution. To access the information from this table we will use the function $E_h(i)$.

Examining the first table, we can assess the diversity in the current population. The higher the number of different activated neurons the higher the diversity. In contrast to this, the search history table gives us an insight into the course of evolution. If for instance a particular neuron was rarely activated, we can deduce that the associated region was not sufficiently explored by the GA. But, before we get to the analysis phase, we have to find ways to represent the contents of both the SOM and the frequency tables.

[1] The words *unit* and *neuron* are used here in a synonymical way.



(a) (b) (c)

Figure 1: A trained SOM in prototype representation(a), fitness landscape representation(b) and search history representation(c)

4.3 Representing the Search History

Although the procedure outlined above effectively processes the chromosome data set it still causes difficulties to the human user to interpret the results. This is mainly due to the numerical output of the procedure. Interpretation becomes easier if we use visual representation forms.

In figure 1a a trained SOM of 10x10 units can be seen. Training was effectuated with binary chromosomes of 16 bits. Each unit is represented by a pattern of 16 pixel visualized as 4x4 blocks. A pattern denotes the underlying prototype or weight vector of that particular unit. Each black pixel in the pattern refers to a 0 while a white pixel refers to 1 in the weight of the prototype vector. For instance, if a unit is depicted by an entirely white block the corresponding weight vector is '1111111111111111'.

If you take a closer look at the map, you detect that there is an axis of blackish units beginning at the lower left corner and reaching to the upper right corner. The prototype vectors of these units have mostly black pixels. This is due to the fact that in the training process genotypically similar chromosomes are clustered onto neighbouring neurons.

Now, we apply the above SOM to a 16bit 1s-counting (Max1) problem. In this problem, the fitness of an individual equals the number of 1s in its chromosome. We visualize the fitness distribution on the map by assigning each neuron a color according to the fitness of the corresponding weight vector. This yields a low resolution picture of the fitness landscape as can be seen in figure 1b. Good units have a light color while worse ones have darker color. The figure confirms the observation made before. The diagonal axis beginning at the left lower corner can now clearly be seen as a dark cluster of low fitness. We would expect a GA to avoid this region and seek out the higher fitness regions in the other corners.

To check this last assumption, we will overlay the information from the search history table onto the map. The information was gathered during a run of a simple GA on the Max1 problem. In figure 1c one sees a slightly faded version of the prototype oriented representation from 1a. In addition, every activation of a unit in the search history table is plotted as a black dot. It is slightly perturbed for easier identification of multiple activations. Each of these dots represents the projection of an evaluated chromosome onto the map. These additional plots tell us which parts of the search space have been explored by the GA.

In figure 1c it can be observed, that the used GA effectively escaped the regions of low fitness. The search is mostly centered around regions of high fitness in the lower right and the upper left corner of the SOM. Furthermore, it can be observed that a few neurons dominate most of the activity, whereas a large number of neurons in the middle of the map have not even been activated once.

What looks like the optimal search strategy at the first sight is in fact an undesirable behaviour of a GA. Fitness landscapes, GAs are normally applied to, are almost always discontinuous, deceptive or have numerous local optima. Otherwise there are techniques better suited for finding global optima like simple hill-climbing for this example. But there is no way that a GA that does not explore such big parts of the search space and loses its exploratory power this fast can succeed in a difficult task.

Generation 20 Generation 30 Generation 70

Figure 2: Visualization of Genetic Drift

Figure 2 visualizes three different states of a population distribution table during the optimization of a multimodal function. The color coding used in the figures above refers to the activation count of each neuron in the population spread table. The darker the color of a neuron the lower its activation count. We see that in generation 20 two different neurons have a high activation count (visualized as

white pixels). These neurons correspond to the subspaces containing the function's global optima. In generation 30 genetic drift focuses the search on one of the two subspaces (the upper left white unit becomes gray). Finally, in generation 70 we notice that almost all neurons have low activation frequencies except of one single peak. At that time there is no more competition among subspaces on the map. The exploratory power and diversity is lost.

Up to this point we have shown that the population distribution table and the search history table are powerful tools to monitor the search process of EAs. The information is mined by SOMs and stored in the tables while meeting all requirements in terms of computational complexity.

5. ENHANCING THE EVOLUTIONARY SEARCH

Based on the information gained from the analysis and the representations described above one can devise strategies to tackle the problems outlined in section 2.1 and 4.3. Steps taken in GASOM towards this goal is the topic of this section.

If at some instance in time evolution focuses too much on a particular set of individuals and the exploratory power is at risk, we can try to guide it towards unexplored regions of the solution space. By looking up the search history table, the least explored subspaces can be easily determined. In subsection 5.1 it will be discussed how the tables holding activation frequencies can be used to develop a *novelty* promoting fitness evaluation. This new fitness evaluation will encourage the exploration of previously neglected solutions. However, this alone will not be enough to tackle premature convergence. As already motivated in subsection 2.1 a powerful operator is needed in order to reintroduce fresh and potential genetic material. This will be explained in subsection 5.2. In subsection 5.3 the question of exploration vs. exploitation will be addressed. A rough measurement for the ratio of exploration and exploitation is introduced and used to balance the evolutionary search.

5.1 Adapting the Fitness Evaluation

In GASOM the fitness of a chromosome is computed by the use of two ranks. The first rank results from ordering the population with respect to the objective value (fitness). The second rank orders individuals by their *novelty*. Novelty here refers to the number of similar chromosomes already encountered during evolution. To determine the novelty factor of a chromosome we use the following function:

$$novelty(c) = 1/E_h(b_c) \tag{5}$$

where c is the current chromosome and b_c is its BMU on the trained Self-Organizing Map. Put in words, novelty is the inverse of the activation frequency in the search history table. The sum of both ranks is then used as the individual's fitness score.

By applying the latter fitness assignment scheme an individual is given two chances to survive in the population. The first possibility is a good performance on the posed (optimization) problem. The second possibility a contribution to the exploration of new subspaces. In phases of high exploitation the activation frequencies of exploited regions will increase drastically. As a result the novelty factor of the corresponding individuals will decrease. Many of them will then vanish from the population, giving place to individuals with higher novelty. Thus, an increase in exploitation will also effect an increase in exploration.

5.2 Confronting Genetic Drift with Reseeding

To regain lost diversity some GA variants employ reseeding operations. For instance, the CHC algorithm [2] reseeds a set of random individuals to restore variation in highly converged populations. In [11] the reseeding operation is based on previously encountered high fitness points.

GASOM performs reseeding by introducing individuals with a high novelty factor into the population. Such individuals correspond to the neurons with low activation count in the search history table. Reseeding involves two steps. First, a so called *reseeding-pool* is created. The reseeding-pool consists of a set of buckets, each of which stands for a particular neuron on the SOM. The buckets are filled by randomly created new chromosomes. Each newly created chromosome is assigned to the bucket which represents its BMU. The requirements we have set up for SOMs in section 4.1 guarantee that the buckets will be uniformly filled. Then, the neurons with lowest activation frequencies are determined. If needed, individuals are sampled (without replacement) from the respective buckets and inserted into the population. To keep the population size constant the individual from the population with the lowest fitness has to be killed instead. Metaphorically speaking the GASOM reseeding operator could be viewed as a resettlement of individuals with a low fitness from overcrowded areas to sparsely populated regions.

To evaluate the performance of this approach, we conducted a few experiments on a 16bit Max1 function. In these tests we compared it to other strategies, namely random-reseeding and reseeding of old individuals. A standard generational GA was used with a population size of 100 individuals. At each generation 10 individuals were requested from the reseeding operator. To assess their quality we compared them with other chromosomes, but did not insert them into the population.

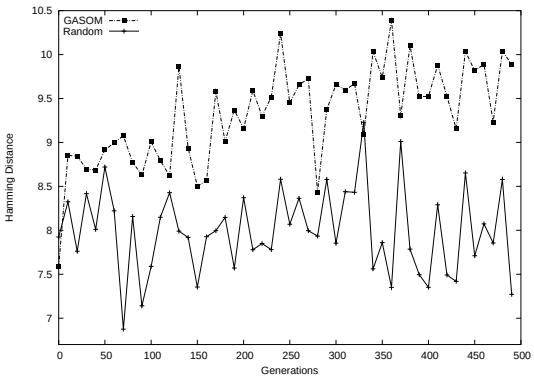


Figure 3: Average Hamming distance of a reseeded individual to all previously evaluated individuals

In figure 3 a plot of the average Hamming distance of reseeded chromosomes to all previously encountered chromosomes can be seen. Random-reseeding maintains an average distance of approximately 8 bits throughout all of the 500 generations. The SOM based reseeding starts with similar values in the first few generations. The average distance

then steadily increases up to approximately 10 bits . This phenomenon occurs due to the fact that the quality of reseeding becomes better the more activations are stored in the search history table.

Table 1: Occurrence number of minimal Hamming distance (reseeded vs. all encountered individuals)

Method	Hamming Distance in Bits				
	0	1	2	3	>3
Old Indiv.	500	0	0	0	0
Random	199	275	25	1	0
SOM	6	241	246	7	0

In the next experiment at each generation the minimal Hamming distance between the reseeded individuals and the encountered individuals is measured. Table 1 shows the number of times each distance was measured. Of course reseeding of old individuals always results in a minimal Hamming distance of 0, as the reseeded individuals are all previously seen. A more sensible comparison with respect to novelty can be done by comparing random- and SOM-reseeding. While the random strategy reseeded in 199 generations a previously seen chromosome, the SOM based strategy did so in only 6 generations. The table also shows, that our new approach reseeded most of the time chromosomes which have at least a Hamming distance of 2 bits to all previously encountered chromosomes.

Table 2: Occurence number of minimal Hamming distance (reseeded vs. population individuals)

Method	Hamming Distance in Bits						
	0	1	2	3	4	5	> 6
Old Indiv.	173	183	104	24	9	8	1
Random	7	55	230	185	23	0	0
SOM	0	0	30	134	256	74	6

The same experiment was then repeated (see table 2). This time however, we measured the Hamming distance between the reseeded individuals and the individuals in the current population. The results confirm our previous observations. The SOM based reseeding never reintroduced an individual which was already in the population, while the other techniques did so in 7 generations (random) and 173 generations (old Indiv.).

We can conclude that the SOM-reseeding strategy effectively maintains high exploratory power. Please note that although reseeding of old individuals performed poorly in the above tests it has the positive side effect of keeping the number of fitness evaluations low.

5.3 Balancing Exploration and Exploitation

The limiting factor of a GA search run is in most cases the numbers of fitness evaluation. Fitness evaluations consume a lot of time in real world application and might even involve testing by an expert. Thus, the goal of every GA should be to get the best results with regard to a limited number of fitness evaluations. The key for an efficient search is the balance between exploration and exploitation. In the beginning of the search premature convergence should be avoided before having covered as much of the search space as possible. In this phase exploration must be enforced while at the end of the search process it is favorably to make the most out of the already found best solutions. So exploitation is the better choice. In other words, the degree of exploitation should be monotonically decreasing and the degree of exploration should be monotonically increasing during the search run, respectively.

The novelty based fitness evaluation, as pointed out in section 5.1, is a good starting point for balancing exploration and exploitation. So far the two ranks are equally weighted to produce the final fitness of an individual. This results in an implicitly well balanced exploration and exploitation. If a dynamic behaviour with high exploration in the beginning and high exploitation in the end is aspired explicitly the ranks could be weighted accordingly. This concept is already implemented for the reseeding operator (see section 5.2). So, how many individuals should be reseeded in a specific time step?

First, the amount of exploration in the current state of the GA is assessed by counting the number of different activated neurons in the population distribution table. Second, a *balance function* is used to determine the number of neurons which ideally should be activated in the current state of the search process. This optimal number of activated units r is calculated by

$$r = \left(1 - \frac{n}{n_{max}}\right) * ||U|| \quad (6)$$

where n is the number of fitness evaluations already performed, n_{max} is the maximal number of fitness evaluations and $||U||$ is the total number of neurons of the SOM. The reseed fraction equals the difference between the ideal and measured neuron count.

Only through the effective combination of both the new fitness assignment scheme and the SOM reseeding operator an efficient exploration strategy can be achieved. If the reseeding operator would be used alone, the introduced new individuals would in most cases die immediately and disapear from the population because novelty is not taken into account. On the other hand, by actively generating novel chromosomes one does not have to rely on crossover and mutation that only by chance might produce new genetic material.

6. EVALUATION AND RESULTS

GASOM was evaluated using three well known problems, namely a Royal Road function [10], the Deceptive F9 function from [9] and the generalized Rosenbrock's function[5]. The chromosome length was 64 bits, 24 bits and 60 bits respectively.

The following parameter setting was used in all test runs. A mutation rate $p_m = 0.005$ and a crossover rate of $p_c = 0.7$ with two point crossover. Selection was accomplished by binary tournament selection. The population size was set to 50 individuals while the maximum number of fitness evaluations was 50.000 evaluations. To reduce statistical noise, each function was run 100 times and the results were averaged.

First a set of experiments was conducted, in which the contribution of the constituent parts of GASOM to the overall performance was assessed. The performance measure used is the average of the best fitness found at the end of each run. Table 3 shows the results of this experiments.

Table 3: Average best fitness of GASOM variants

Method	Deceptive F9	Royal Road	Rosenbrock
Novelty	24,19	167,20	0,02575
Reseeding	28,28	185,76	0,00119
Nov. & Res.	**30,00**	**220,56**	**0,00028**

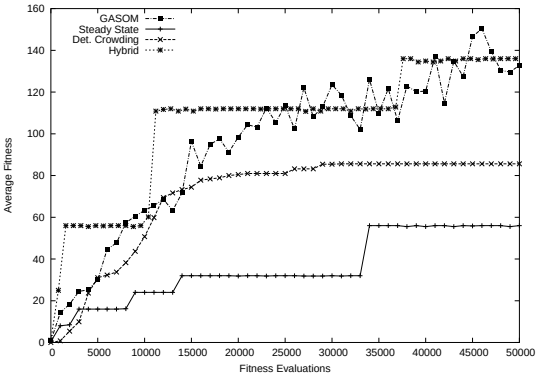


Figure 4: Performance on Royal Road

Best results are highlighted in bold. It can clearly be seen, that the best performance is achieved when novelty and and reseeding are used in combination. The synergetic effect between the new fitness assignment scheme and the reseeding operator leads to a significant improvement. This confirms the assumption made in section 5.3 that both proposed elements are needed for an efficient exploration strategy.

Next, another set of experiments was conducted in which we compared the performance of GASOM to a standard steady-state GA, deterministic crowding [9] and the novel hybrid replacement scheme of Lozano et al. [8]. These variants have already been shown to significantly improve the performance of evolutionary search. Table 4 shows the results of this experiment.

Table 4: Average fitness of different GAs

Method	Deceptive F9	Royal Road	Rosenbrock
SteadyState	24,00	95,44	0,08394
Det. Crowding	26,88	188,72	0,00067
Hybrid	22,54	**227,20**	0,09671
GASOM	**30,00**	220,56	**0,00028**

Both, on the Deceptive F8 function and the Rosenbrock function GASOM yields the best results out of the tested GAs. On the Royal Road function the hybrid method performed slightly better. However, its performance deteriorates on the other two functions. We may observe that GASOM achieves a good trade-off in performance on different problems and gives more stable results. Due to its improved exploration ability, GASOM is able to drive the evolution towards the most promising new regions.

Figure 4 shows plots of the average fitness during a run of the Royal Road function. Both the simple steady-state GA and the GA with hybrid replacement exhibit the typical fitness jumps as new building blocks are rapidly propagated among population individuals. Deterministic crowding has a very neat looking monotic behavior but convergences prematurely. In GASOM the average fitness makes small oscillations while increasing in a nearly linear fashion. The oscillations are due to the interplay between the new exploration techniques and the greediness of the GA.

Table 5: Number of times global optimum was found (out of 100)

Method	Deceptive F9	Royal Road	Rosenbrock
SteadyState	0	6	0
Det. Crowding	44	50	0
Hybrid	3	**77**	0
GASOM	**100**	72	0

Table 5 shows the number of times the global optimum was found on the used test functions. Our method achieves outstanding results on the Deceptive F9 function. In all 100 runs it was able to find a global optimum. The runner-up on this function (Deterministic Crowding) was able to find the global optimum in 44 times, while the other GAs did so in less than 5 times. The intelligent exploration scheme significantly improved the efficiency of the search process on deceptive functions. This was confirmed by other experiments we conducted. The analysis of the search history table gave insight into why GASOM performed better on this type of problems. Typically, on deceptive problems the search is lead away from the global optimum. Consequently, the corresponding units in the search history table will be activated less. The reseeding operator reintroduces individuals from this part of the search space, while the novelty based fitness assignment scheme gives them a chance to survive and reproduce.

Although diversity is not the primary goal of our proposed GA, it should implicitly lead to a balance of diversity. In order to measure the population diversity we used the following function[3]:

$$d(P) = \frac{\sum_{i=1}^{L} min(F_i, 1 - F_i)}{L/2} \tag{7}$$

and

$$F_i = \frac{\sum_{j=1}^{N} P_i(j)}{N} \tag{8}$$

where N equals the population size and L is the chromosome length in bits. $P_i(j)$ returns the allele at the j^{th} gene of the chromosome. Figure 5 shows the change in diversity during an optimizationn run of the Royal Road function. It can be seen, that GASOM effectively balances the diversity according to the phase of evolution. In the beginning it maintains high diversity, which then linearly drops to nearly 0 in the later phases of exploitation.

7. CONCLUSIONS AND FUTURE WORK

In this work we proposed an intelligent exploration technique for GAs. The approach uses SOMs to analyize data of the evolution process and utilize it to enhance the search strategy. This lead to the development of a GA with four new core components: an explicit representation of the search history, a fitness evaluation promoting novelty, a reseeding operator preserving exploratory power and a control mechanism balancing exploration and exploitation. This framework, called GASOM, was tested against several other ap-

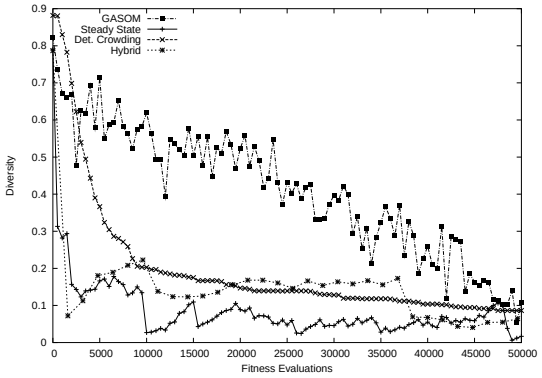


Figure 5: Diversity on Royal Road

proaches and proved to be well suited for preventing premature convergence.

GASOM performed considerably well on deceptive problems where other GAs tend to get stuck in local optima. The employed reseeding operator succesfully resisted genetic drift. Hence, exploratory power was maintained where necessary. Besides that the SOM offers a neat tool for visualizing the state of evolution.

An obvious refinement of the discussed algorithm would be an explicit adaption of the fitness assignment. This could be achieved by weighting the influence of the novelty rank according to the measured degree of exploration (see section 5.3). Another starting point for straight forward fine-tuning is the balance function. Up to now only a linearly decreasing function was used. Exponential decline might be more effective. Another minor adjustment could make GASOM find as many different optima as possible instead of focusing on one global optimum. Once more the key would be an intelligent compromise between exploration and exploitation.

A more fundamental issue is the size of the SOM. The effect of varying the number of neurons on the performance has not been investigated so far. An optimal trade-off between accuracy of the search history table and computational demands is desirable.

Furthermore, we aim at adjusting GASOM to multi-objective problems. A promising study could be the identification of each unit's part of the Pareto front. The same methods used in GASOM for balancing exploration and exploitation (see section 5.3) could be used to balance the different objective functions, accordingly. For this task other dimensionality reduction techniques might be more appropriate than SOMs.

Finally, we plan to apply GASOM in machine learning environments. More specifically we are working on its application to robot control in the RoboCup domain.

8. ACKNOWLEDGMENTS.

We would like to thank our advisors Oliver Obst and Jan Murray for their encouragement and the stimulating discussions. Thanks to Thomas Kleemann for his insightful hints and comments. We are also grateful to the anonymous reviewers for their constructive and helpful comments.

The authors are partially supported by the German research foundation *DFG*.

9. REFERENCES

[1] L. Eschelman and J. Schaffer. Preventing premature convergence in genetic algorithms by preventing incest. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 115–122. Morgan Kaufmann, 1991.

[2] L. Eshelman. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms*, pages 256–283. Morgan Kaufmann, 1991.

[3] C. Fernandes and A. Rosa. A study on non-random mating and varying population size in genetic algorithms using a royal road function. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 60–66. IEEE Press, 2001.

[4] D. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transaction on Neural Networks*, 5(1):3–14, 1994.

[5] D. A. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[6] J. Hu, K. Seo, Z. Fan, R. Rosenberg, and E. Goodman. Hemo: A sustainable multi-objective evolutionary optimization framework. In *Proc. 2003 Genetic and Evolutionary Computing Conference*. Springer Verlag, July 2003.

[7] T. Kohonen, T. Kohonen, M. R. Schroeder, and T. S. Huang. *Self-Organizing Maps*. Springer-Verlag New York, Inc., 2001.

[8] M. Lozano, F. Herrera, and J.R.Cano. Replacement strategies to preserve useful diversity in steady-state genetic algorithms. *In Press*, March 2004.

[9] S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1995.

[10] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous System: Proceedings of the First European Conference on Artificial Life*, pages 245–254, 1991.

[11] K. Rasheed. *GADO: A Genetic Algorithm for Continuous Design Optimization*. PhD thesis, Rutgers University, New Brunswick, NJ, 1998.

[12] G. Romero, J.J.Merelo, P. Castillo, J.G.Castellano, and M. Arenas. Genetic algorithm visualization using self-organizing maps. In *Parallel Problem Solving from Nature - PPSN VII*, pages 442–451. Springer Verlag, 2002.

[13] E. Ronald. When selection meets seduction. In *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995.

[14] B. Sareni and L. Kraehenbuehl. Fitness sharing and niching methods revisited. *IEEE Transaction on Evolutionary Computation*, 2(3):97–106, September 1998.