# Genetic Algorithms, Clustering, and the Breaking of Symmetry

**Martin Pelikan and David E. Goldberg**

IlliGAL Report No. 2000013
March 2000

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-0897
Fax: (217) 244-5705

# Genetic Algorithms, Clustering, and the Breaking of Symmetry

**Martin Pelikan and David E. Goldberg**
Illinois Genetic Algorithms Laboratory
Department of General Engineering
University of Illinois at Urbana-Champaign
{pelikan,deg}@illigal.ge.uiuc.edu

March 11, 2000

**Abstract**

This paper introduces clustering as a tool to improve the effects of recombination and incorporate niching in evolutionary algorithms. Instead of processing the entire set of parent solutions, the set is first clustered and the solutions in each of the clusters are processed separately. This alleviates the problem of symmetry which is often a major difficulty of many evolutionary algorithms in combinatorial optimization. Furthermore, it incorporates niching into genetic algorithms and, for the first time, the probabilistic model-building genetic algorithms. The dynamics and performance of the proposed method are illustrated on example problems.

## 1 Introduction

Symmetry is one of the major difficulties of genetic algorithms on combinatorial problems. In spite of that the nature of the problem encourages the use of some kind of recombination, due to the symmetry of the problem the recombination often slows down the convergence by disrupting good solutions. Moreover, the use of niching in the probabilistic-model building genetic algorithms was identified as an important issue when extending the algorithms to solve difficult hierarchically decomposable problems (Pelikan, Goldberg, & Cantú-Paz, 2000).

In this paper we propose a method to both deal with the symmetry in a problem as well as incorporate niching. The method is not intended for any particular recombination operator and can be used to extend and improve the recombination in both the simple genetic algorithms as well as the probabilistic model-building genetic algorithms. Each generation the set of selected parents is clustered according to their genotypes. Each cluster is then processed separately, yielding some offspring. The number of offspring produced by each cluster can be either proportional to its size or its average fitness. By avoiding juxtaposition of the solutions from different clusters, the negative effect of symmetry in a problem is alleviated. By assigning each cluster resources proportional to its average fitness, niching is incorporated to allow the algorithms to thoroughly search the solution space for multiple optima. Even though niching is a widely studied concept in genetic algorithms, this is the first attempt to use it in the probabilistic model-building genetic algorithms.

The paper starts by a brief introduction into genetic algorithms and the problem of symmetry. Section 4 introduces the class of clustering evolutionary algorithms that process the population of selected parents in separated clusters. The k-means clustering method that was used to cluster the

parent population in our experiments is introduced in Section 5. Section 6 presents the results of our experiments. The paper is summarized and concluded in Section 7.

## 2 Genetic Algorithms

A genetic algorithm (GA) (Holland, 1975; Goldberg, 1989) evolves a population of potential solutions to a given problem. The first population of solutions is generated at random. By means of a measure of quality of solutions given by a user, usually expressed in the form of one or multiple functions, better solutions are selected from the current population. The selected solutions undergo the operators of mutation and crossover (recombination) in order to create the population of new solutions (the offspring population) that fully or in part replace the original (parent) population. The process repeats until the termination criteria (e.g., convergence to a singleton) given by the user are met.

Probabilistic model-building genetic algorithms (PMBGAs), also called the estimation of distribution algorithms (Mühlenbein & Paaß, 1996), replace genetic two-parent recombination of the genetic algorithms by building an explicit macroscopic model of promising solutions and using the constructed model to guide the further search. As models, probability distributions are used. For an overview of recent work on PMBGAs, see Pelikan, Goldberg, and Lobo (1999).

In this paper we discuss only a simple univariate marginal distribution algorithm (UMDA) that uses a fixed distribution estimate which assumes that the variables in a problem are independent. The value of each variable is determined only by its distribution in the set of selected parents without taking into account the contexts in which these values were present. This makes the algorithm work very well on linear and other problems without significant interactions of variables.

## 3 Symmetry

For many combinatorial problems there are a number of different solutions to a problem. Moreover, many regularities in the entire landscape can often be observed. In a graph bisection, for instance, the goal is to partition the nodes of a given graph into two equally sized groups so that the number of edges between the groups is minimized. Each bit in the solution string corresponds to one node in the graph and its value determines the group to which this node is assigned. It is easy to see that in this problem, there are at least two optima which are complementary. Moreover, the average fitness of any schema is equal to the average fitness of the complement of the schema which is fixed in the same positions as the original schema but to the exactly opposite values, e.g.

$$f(***00*1**) = f(***11*0**).$$

This implies that the fitness of each solution does not depend on the value of a particular bit or a set of bits but on the overall combination which can be often difficult to obtain. Each schema and its complement have the same fitness on average and unless the population drifts to either side, the GAs have no mechanisms to decide which way to go from a uniformly distributed population.

Both the GAs as well as the PMBGAs guide the exploration of the search space to regions that can be reached by combining important parts of promising solutions found so far. However, in case of symmetric problems, this often yields to a decrease in the solution quality. In the simplest case (e.g., the graph partitioning mentioned above), there are two complementary parts of the search space that are to be explored. However, combining high-quality solutions and their complements which are equally good often results in poor solutions. Furthermore, as it was pointed out above, the algorithm has no means of deciding between complementary partial solutions since both seem to be of the same quality on average. If the niching were incorporated to eliminate genetic drift,

the GAs would either converge very slowly or would never reach the optimum. This becomes a crucial problem for the PMBGAs that use only macroscopic information about the partial solutions in the population of parents to generate new offspring. The problem can be eliminated only by using more complex models that would take into account higher order dependencies. With a more complex model, traditional niching methods as tournament selection with continuous sharing could be used. However, using more complex models results in extra computational resources required to find such model and we would prefer to use the simplest model we can.

Similar property can be observed in a simple symmetrical two-max function with equally sized peaks which is defined as

$$f_{two-max}(X) = \left| \frac{n}{2} - u \right|, \tag{1}$$

where $u$ is the sum of bits in the input string, $n$ is the length of the input string, and $|\cdot|$ denotes absolute value. This function has two global maxima in $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$, and the fitness of each solution is equal to the fitness of its complement. Even though the two-max is composed of two simple linear functions which are very easy to optimize by most evolutionary methods, their convergence on the two-max can get very slow.

The problem of symmetry was also studied on spin-glass systems in Naudts and Naudts (1998) and other problems, in all of which the simple genetic algorithms and other algorithms with the same bias experience great difficulties. However, the class of problems where we encounter symmetries in some form (not necessarily "clean" and "perfect" symmetry) contains many interesting and important real-world problems that could benefit from recombination. The question is how to resolve this problem without having to design special coding, heuristics, or other problem-specific operators. In the next section we propose a method that can be used in many evolutionary algorithms to alleviate the problem of symmetry and consequently improve their performance. Additionally, the method eliminates the source of difficulties of using niching in the PMBGAs and is the first attempt to address niching in the PMBGAs. Other niching methods can be used in concert with our method, and the method can be extended to also affect the selection.

## 4 Improving Recombination by Clustering

In all problems mentioned above there are two complementary parts of the search space, each with the same structure. This structure can be very simple as in the two-max function where both parts are simple linear unimodal functions or more complex as in the graph partitioning where in most cases each part contains a large number of local optima. However, there exist algorithms that are able to deal with a wide range of problems and if they were able to distinguish between the two parts of the solution space, they would be able to optimize the problem very efficiently. The motivation to introduce clustering in evolutionary algorithms is that by helping the algorithm to separate the two or more complementary parts of the solution space, the problem of symmetry would be eliminated and the algorithms would simply not have to deal with it. By using algorithms which can solve the problem if the symmetry is not present in a problem (as a linear problem in case of two-max), the problems could be solved very efficiently, accurately, and reliably.

Recently, clustering was used in a businessman-customer scheme for an effective niching in genetic algorithms (Goldberg & Wang, 1997). It has proven to be a very powerful method for discovering and maintaining solutions close to a number of different optima. We use a similar scheme; however, clustering is not used only to improve niching while selecting better solutions from the entire population, but also to separate unlike parts of the search space and process each part separately. Furthermore, clustering is not directly driven by fitness but the genotype itself.

A similar concept of using multiple populations, each corresponding to one optimum (in ideal case), was introduced by Hocaoğlu and Sanderson (1995). The above work resulted in the proposal of the minimal representation size cluster genetic algorithm (MRSCGA). The MRSCGA starts with one intact population. After a couple of generations the population is clustered into a number of sub-populations by using the minimal representation criterion (Segen & Sanderson, 1981). Each sub-population is then processed separately for a number of generations. Subsequently, the statistics over the entire population is computed to get a new partitioning of the population, and the process is repeated, allowing each sub-population more and more generations as the time goes on. The MRSCGA was designed for dealing with multimodal functions and not to directly address the problem of symmetry. The issues of recombination and the effect of clustering to improve its power were not addressed. Moreover, the use of niching within this framework was not discussed and it was assumed that the niching comes up naturally by using multiple separated sub-populations. This is often not the case with finite populations where noise is an important factor affecting the convergence properties. In our model we address niching where the resource allocation is guided by solution quality.

Another context where the evolutionary computation and clustering are combined looks at the problem from the opposite side and tries to use robust evolutionary optimization methods to cluster a given sample (Bezdek, Boggavarapu, Hall, & Bensaid, 1994).

Melhuish and Fogarty (1994) use restricted mating to search the state space in separate regions for an efficient learning of classifiers. Restricted mating selects and processes classifiers that best agree with the underlying state space model by picking a random region of the search space. Recently, similarities among parents were also used to improve various selection and niching schemes (Deb, 1991; Horton, 1995; Harik, 1995).

In our framework the general algorithm proceeds as follows. The initial population is generated at random just like in the simple GAs. Each iteration of the main loop, the better solutions are selected from the current population. Before the recombination proceeds, the set of selected solutions is partitioned into a number of clusters. The number of clusters can be either given on input or determined by using hierarchical clustering methods or the minimal representation criterion (Segen & Sanderson, 1981). Recombination proceeds in each cluster separately and produces a number of new individuals, the offspring. Any recombination can be used, e.g. two-parent crossover of simple genetic algorithms, forward simulation of a probabilistic model learned by the Bayesian optimization algorithm, etc. The number of offspring produced by each cluster can be either proportional to its size or its average fitness which introduces niching and assigns each cluster resources proportional to its overall quality. The offspring are then incorporated into the original population, possibly replacing the entire population. The loop finishes when the termination criteria given by the user (e.g., convergence, maximum number of generations, etc.) are reached. The pseudo-code of the algorithm follows:

(1) $t \leftarrow 0$
(2) Randomly generate initial population $P(0)$.
(3) Select a set of promising strings $S(t)$ from $P(t)$.
(4) Cluster $S(t)$ into $k$ clusters $C_i(t)$.
(5) Process each cluster $C_i(t)$ separately to generate its offspring $O_i(t)$.
(6) Create a new population $P(t+1)$ by replacing some strings from $P(t)$ with $O_i(t)$.
(7) Set $t \leftarrow t+1$.
(8) If the termination criteria are not met, go to 3.

As it was pointed out above, the contribution of the proposed framework is twofold: (1) the negative effect of symmetry in a problem is alleviated, and (2) the use of effective niching in the PMBGAs is allowed. In the PMBGAs, the use of traditional niching methods often fails to achieve the goal and results in a very poor performance. Once niching can be incorporated into the PMBGAs, it can be used to improve their performance on difficult combinatorial problems, solve hierarchical problems (Pelikan, Goldberg, & Cantú-Paz, 2000), and tackle multi-objective problems by thoroughly searching the solution space for a diverse Pareto front. It is important to note that the effects of symmetry can be eliminated by using more complex models. However, the more complex the models get, the more resources it takes to learn them. Thus, avoiding using complex models whenever we can seems to be a step in the right direction.

## 5 k-Means Clustering

In k-means clustering, each cluster is specified by its *center*. Initially, $k$ centers (where $k$ is given) are generated at random. Each point is assigned to its nearest center. Subsequently, each center is recalculated to be the mean of the points assigned to this center. The points are then reassigned to the nearest center and the process of recalculating the centers and reassigning the points is repeated until no points change their location after updating the centers. The pseudo-code of the k-means clustering algorithm follows:

(1) Generate $k$ centers at random.
(2) Assign each point to the nearest center.
(3) Move each center to the mean of the points assigned to it.
(4) If point locations have changed in step 2, go to 2.
(5) Return the cluster centers and point locations.

To cluster binary strings, we can simply use real vectors of the same length to represent the center of each cluster. Euclidean metric can be used to measure distance. Alternatively, phenotypic distance can be used to cluster the population which can be very useful on real-valued problems. In this case the centers can be also updated by computing frequency of each bit on each position and fixing each position of the genotype of the center to the most frequent value on this position. The value of the center would then be its phenotype.

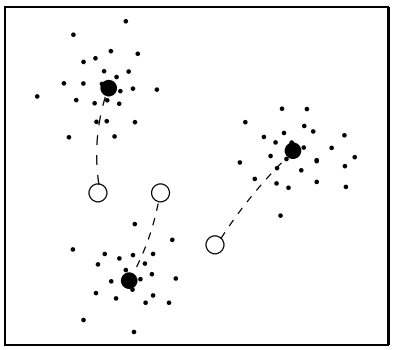

Figure 1: Example run of a k-means clustering method.

The clusters can be also adjusted on the fly as the point locations are being updated which speeds up the computation slightly. The initialization of cluster centers can be improved by assigning each center to a randomly chosen point or the mean of a sample of points drawn randomly from the population that is to be clustered. In our implementation we initialize each center to a randomly picked solution. An example of the set of points in two-dimensional space, the initial position of the centers, and their final positions, are shown in Figure 1. The initial centers are displayed as empty circles, the trajectory of each center is represented by the dashed line, and the final positions of the centers are shown as full circles.

The distance metric used in the clustering algorithm is also a very important issue and for very complex problems this may lead to anomalous results. In general, the more similar the genotype metric is to its phenotype equivalent, the better the clustering should work. Phenotype distance metric can also be used, if possible, as it was discussed above.

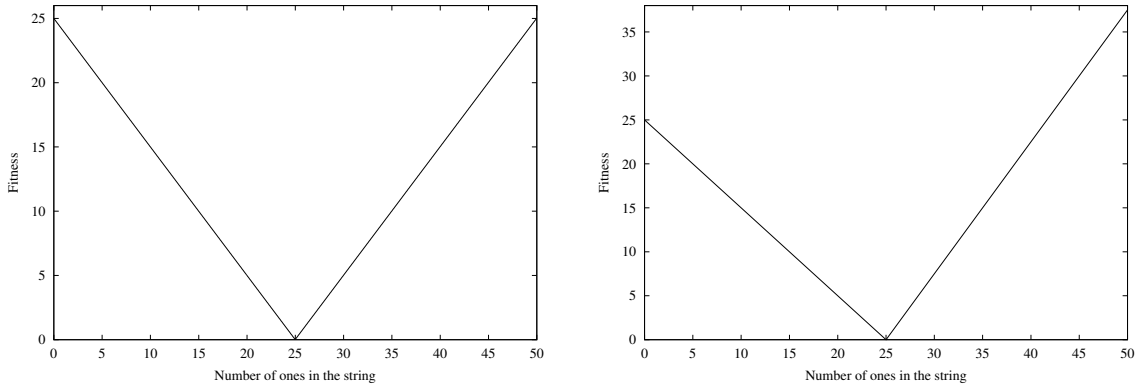


Figure 2: Twomax with a) equal and b) unequal peaks (where $f(1\ldots1) = 1.5f(0\ldots0)$).

More sophisticated clustering methods can be used. We have chosen k-means clustering for its simplicity. We believe that the use of better clustering methods will lead to better results. However, we must keep an eye on the complexity of the used clustering method not to lose more than we gain.

# 6 Experiments

We have performed various experiments to demonstrate the effect of clustering on recombination. We have implemented clustering in two algorithms—the simple genetic algorithm with one-point crossover and the univariate marginal distribution algorithm. This study considers four problems: (1) two-max with equal peaks, (2) two-max with unequal peaks, (3) graph-bisection, and (4) Ising spin-glass systems. The two-max problem is very simple and was chosen to demonstrate ideal behavior of clustering. A similar behavior should be observed on more difficult problems by using more sophisticated operators together with clustering. Two cases were chosen in order to show the effects of niching. The remaining problems were chosen to demonstrate the power of clustering on highly multimodal problems with strong symmetries that are known to be very difficult for the simple genetic algorithms without using additional problem-specific operators dealing with symmetry.

The first problem is a simple two-max function which is a very good example to show the expected behavior of the extended algorithms. Even though most interesting problems are not as simple as the two-max, by using more sophisticated recombination operators similar behavior as the one of the simple UMDA on two-max can be observed. We used two two-max functions, one with equal peaks (see Equation 1 and Figure 2a), and one with one peak higher than the other by a factor of 1.5 (see Figure 2b), defined as

$$f^*_{two-max}(X) = \begin{cases} \frac{n}{2} - u & \text{if } u \leq \frac{n}{2} \\ 1.5\left(u - \frac{n}{2}\right) & \text{otherwise} \end{cases}.$$

The dynamics of the UMDA with two clusters on the two-max problems with equal peaks of size $n = 50$ is shown in Figure 3. Histograms with the number of ones in a solution on the horizontal axis and the number of corresponding solutions on the vertical axis are shown in a number of generations. The optima are on the left-most and right-most sides of the histograms. The size of the population was $N = 500$, binary tournament selection was used, and the offspring

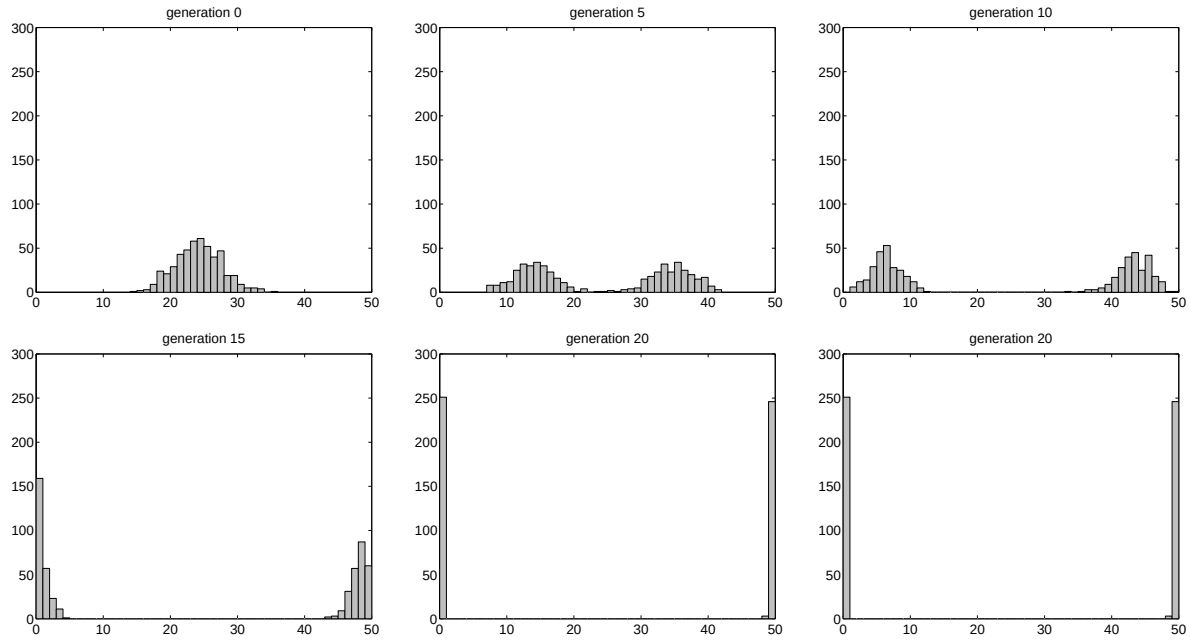


Figure 3: Dynamics of the UMDA with two clusters on two-max with equal peaks of size $n = 50$. Generations 0, 5, 10, 15, and 20 are displayed from left to right.

replaced entire original population. Initially, most of the mass is centered around the middle with the binomial distribution of mean $\mu = \frac{n}{2}$ which follows from that the initial population is generated at random with a uniform distribution and the number of solutions with $l$ ones is equal to $\binom{n}{l}$. As the selection proceeds, it's pushing the population to both left and right sides. The clusters capture the division and each of them follows its optimum. The size of offspring generated by each cluster is proportional to its fitness, and thus the solutions are equally distributed between the two optima.

If the population were not separated into clusters each time we select parents, pressure from both sides would make it harder for the algorithm to converge to either side. Eventually, the population would drift and the algorithm would converge to one optimum. However, only one optimum would be found. If niching were used to eliminate drift and search for multiple solutions, the UMDA would require exponential time to converge to the optimum. Clustering allows the population to converge to the both optima very quickly and reliably. Another important point is that if we allowed generation gap and would keep best individuals of the original population, the lower optimum would be eliminated and all individuals would concentrate around the higher optimum. The effect of symmetry could also be alleviated by using more complex models; however, there is no reason to use complex models where simple models are sufficient.

Analogous experiment with the two-max with unequally sized peaks of size $n = 50$ is shown in Figure 4. As we can see, the algorithm finally discovers both optima and distributes individuals to each of these proportionally to their fitness. An interesting behavior can be observed in generation 20 (middle part of Figure 4). Even though more individuals are distributed in the neighborhood of the higher optimum, the algorithm converges faster to the lower one. This seems to be an effect of the tournament selection. At certain generation, the fitness of any individual close to the higher optimum is higher than the fitness of any individual close to the lower optimum. Then,

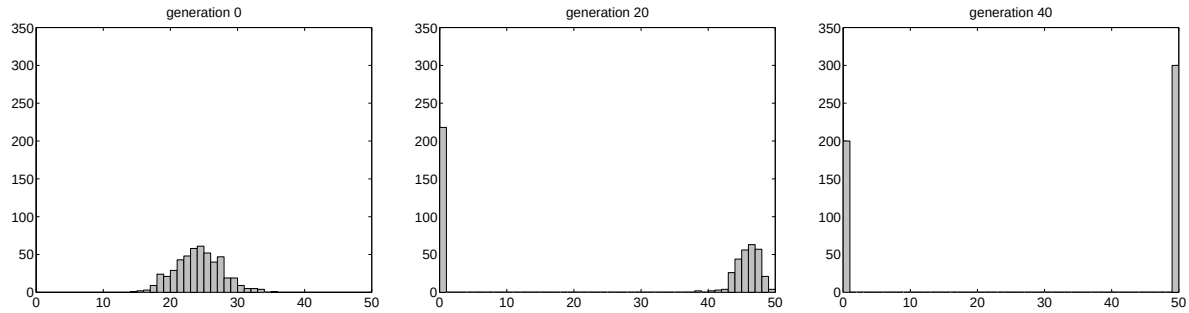


Figure 4: Dynamics of the UMDA with two clusters on two-max with $f(1\ldots1) = 1.5f(0\ldots0)$. Generations 0,20,and 40 are displayed from left to right.

the only chance to win tournament for the solutions close to the lower optimum is to compete with a worse solution from the same part. Thus, all winners from the lower basin of attraction are selected in competition with other members of this basin, and the pressure on their fitness is as if the sub-population on this basin was processed separately. However, the winners on the opposite basin often win only because they were competing with someone from the lower basin and only part of the time they compete with each other. This decreases the selection pressure on their fitness values, and the corresponding cluster converges slower.

It is important to note that if we used too strong selection, e.g. truncation selection with $\tau = 50\%$, we would eliminate the lower optimum before the population would converge to the both optima because even if we discovered both optima, the proportion of the higher optimum would be about 60% of the population (proportional to its fitness). Truncation selection would then select only solutions at the higher peak. Thus, to have an effective niching, one must assure that the selection pressure is not too strong. This is an important issue of niching in general. Another important note is that the two-max problem with unequal peaks is not strongly affected by the problem of symmetry and the UMDA without clustering would be able to find the global optimum (and loose the local one).

We also performed some experiments on the graph bisection problem of a two-dimensional grid cut in halves which are connected by only two edges (Schwarz & Ocenasek, 1999). Each position in a string corresponds to one node in the graph and its value determines the classification of the corresponding node to either of the two classes. This problem requires a repair operator since only solutions with equal numbers of zeroes and ones are allowed. To repair the solutions, various operators can be used. In order to make our results independent of the ordering of the nodes in the solution strings, we use a randomized repair operator which turns on/off a required number of bits by randomly picking positions and changing the value of a particular bit if it moves the solution closer to a legal one. More sophisticated heuristic operators would improve the convergence significantly; however, this was not the goal of our experiments which were designed to show how powerful clustering can be.

The fitness of each string is determined by the size of the graph minus the number of edges between the two partitions of nodes specified by the string. The goal is to maximize this value. Since there are two edges between the optimal partitions, the optimal fitness is $n-2$, where $n$ is the size of a problem (number of nodes in the graph). We have tried problems of size $n = 16$, $n = 36$, and $n = 64$. In all tested algorithms truncation selection with $\tau = 50\%$ was used and the worst half of the original population was replaced by the offspring.

All problems have shown to be extremely difficult for the simple GA with both uniform and one-point crossover, with or without mutation. The UMDA with an intact population also could not solve the problem. On the other hand, the UMDA algorithm with $k = 2$ clusters converged very fast on problems with $n = 16$ and $n = 36$ even though its equivalent with only one cluster was not able to get better solutions than the ones in the initial population even with huge populations. The average number of evaluations until the population in the UMDA with $k = 2$ clusters got overtaken by optima in 9 of 10 independent runs is shown in Table 1. For a problem of size $n = 64$ and $k = 2$ clusters, the algorithm was often deceived to one of the local optima. Introducing additional two clusters resolved this difficulty and improved the performance of the algorithm. In all problems, the algorithm discovered both global optima which were equally distributed in the final population.

The graph bisection is a very complex problem with many local optima that are very hard to get from once deceived. Even though the simple genetic algorithm with two different recombination operators and the UMDA were not able to solve any of the tested problems by using a reasonable amount of resources, the UMDA with a very simple recombination that assumes the variables in a problem to be independent was able to solve all problems quite efficiently with only a few clusters. This suggests that using clustering for difficult problems with symmetries can alleviate some difficulty and make the problem easier to be solved by even a very simple algorithm. Using more sophisticated algorithms (particularly, the Bayesian optimization algorithm (Pelikan, Goldberg, & Lobo, 1999)) on this problem results in much better scale-up behavior (Schwarz & Ocenasek, 1999). We have tried to apply this algorithm to the problem with and without clustering and for small instances the behavior of the algorithm with clustering was the same or a a little worse than without clustering. This is caused by that the clustering reduces diversity in the population and using Bayesian networks with only interactions of low order is sufficient to solve this problem quite efficiently. We are currently investigating the behavior of the BOA in more detail.

| **Problem size** | **Population size** | **Number of clusters** | **Avg. number of evaluations** |
|---:|---:|---:|---:|
| 16 | 50 | 2 | 406 |
| 32 | 600 | 2 | 11633 |
| 64 | 5500 | 4 | 140555 |

Table 1: Results of the UMDA with clustering on the regular graph bisection.

We also applied the UMDA and simple GA with clustering on one and two-dimensional Ising spin-glass instances. These problems are known to be difficult for GAs with traditional recombination (Pelikan & Mühlenbein, 1999; Naudts & Naudts, 1998); however, they can be solved very efficiently by using advanced recombination operators based on methods from Bayesian networks learning and inference (Pelikan et al., 1998). The problem is to find a binary value for each node in a linear sequence or a grid of nodes correlated with two types of pair-wise functions. One function gives a positive reward to the overall functional value for the connected nodes if their values are the same and a negative reward otherwise. The other returns exactly opposite values.

All algorithms performed quite well on a one-dimensional instance (a chain). However, none of the tested algorithms was able to solve a two-dimensional spin-glass system (a 2D grid) adopted from Pelikan et al. (1998) even with very large populations without significantly increasing selection pressure. In spite of this, a major improvement in convergence of the UMDA was achieved with clustering. Without clustering, the UMDA was not able to improve its initial solutions that were generated at random. The disruption of building blocks became a dominant factor and did not allow the remaining mechanisms to improve upon the initial population. However, when the clustering was used, the algorithm was able to find solutions very close to the optimum (about 162, where

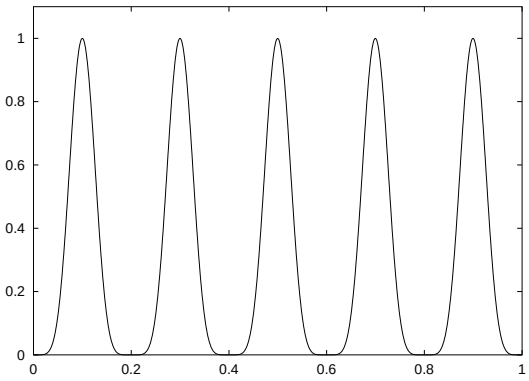

Figure 5: F1 fitness function with 5 equally sized peaks.

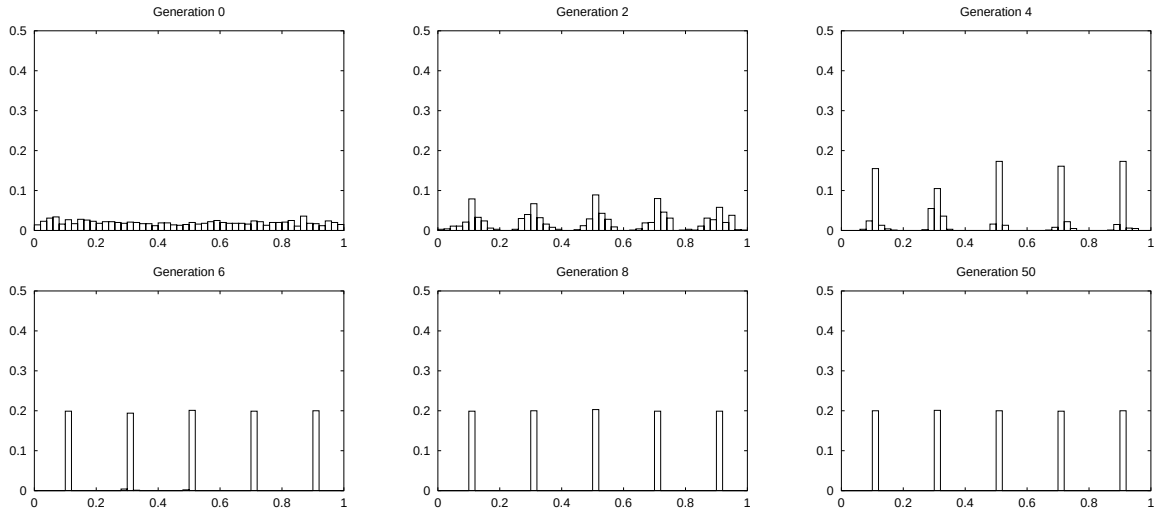


Figure 6: Dynamics of the BOA with five clusters on $F_1$ of size $n = 30$. Generations 0, 2, 4, 6, 8, and 50 are displayed from left to right.

optimum has fitness of 172).

To demonstrate the effects of niching in the probabilistic model-building genetic algorithms, we have performed some experiments on the real-valued function with 5 equally sized peaks shown in Figure 5. The function is adopted from (Mahfoud, 1995), defined as

$$F_1(x) = \sin^6(5\pi x),$$

where $x$ is a real number from interval $[0, 1)$, corresponding to the normalized binary integer value of the input string.

We have used the UMDA with $k = 5$ clusters. The function has shown to be not very difficult to optimize and most of the settings resulted in a very robust performance. In Figure 6, dynamics of the population distribution is shown. The population of size was set to $n = 1000$, binary tournament selection was used, offspring replaced entire parent population, and clustering was run on the phenotype with 25 random restarts. The number of restarts was chosen in order to let the centers hit each basin at least once. The results suggest that the algorithm is able to reach and maintain all

peaks without specifying any thresholds as is usual in other niching methods. Similar results can be obtained on functions with unequal peaks that are non-uniformly distributed. Without clustering or niching, even sophisticated algorithms like the BOA, as well as the simple GAs, are not able to maintain all 5 optima. The UMDA without clustering is, in fact, not able to maintain more than one optimum. However, clustering eliminates some problem difficulty and allows the UMDA reach and maintain all optima very accurately and reliably.

# 7 Summary and Conclusions

The paper proposed the use of clustering in recombination of genetic and evolutionary algorithms with the goal of making the task the recombination is to achieve easier. It provided and discussed a number of interesting results regarding the dynamics and efficiency of the method.

Clustering of the population of parents and processing each cluster separately alleviates the problem of symmetry and makes the job for recombination easier. It can automatically be used to incorporate niching into the algorithm. Simple recombination methods can get much further with than without clustering; however, clustering does not solve the entire problem by itself and more sophisticated operators together with clustering must be used for an efficient scale-up behavior on difficult problems. Assuming we have a powerful enough method for combining promising solutions that is capable of covering important interactions of the decision variables in a problem, clustering can be used very efficiently to thoroughly search the solution space and discover multiple optima.

However, for simple problems the use of clustering results in unnecessary loss of diversity and leads to an increase in an adequate population size. When the model is sufficient to capture the structure of the problem, including its symmetry, clustering needs not be incorporated and other niching methods can be used instead. However, by using clustering on symmetrical problems, simpler models can be used with the same effect.

# Acknowledgments

The authors would like to thank Martin Butz, Erick Cantú-Paz, Bart Naudts, and Dirk Thierens for useful discussions and insights.

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-97-1-0050. Research funding for this work was also provided by the National Science Foundation under grant DMI-9908252. Support was also provided by a grant from the U. S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, the U. S. Army, or the U. S. Government.

# References

Bezdek, J. C., Boggavarapu, S., Hall, L. O., & Bensaid, A. (1994). Genetic algorithm guided clustering. *Proceedings of the First IEEE Conference on Evolutionary Computation*, *2*, 34–39.

Deb, K. (1991). *Binary and floating-point function optimization using messy genetic algorithms.* Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Goldberg, D. E., & Wang, L. (1997). Adaptive niching via coevolutionary sharing. In *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science* (Chapter 2, pp. 21–38). West Sussex, England: John Wiley & Sons Ltd.

Harik, G. R. (1995). Finding multimodal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 24–31.

Hocaoğlu, C., & Sanderson, A. C. (1995). Evolutionary speciation using minimal representation size clustering. *Evolutionary Programming IV*, 187–203.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Horton, B. (1995). *The restricted breed GA for multi-modal genetic algorithms: Comparison with sequential niching, crowding and sharing.* Submitted to Evolutionary Computation.

Mahfoud, S. W. (1995, May). *Niching methods for genetic algorithms.* Doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL. Also IlliGAL Report No. 95001.

Melhuish, C., & Fogarty, T. C. (1994). Applying a restricted mating policy to determine state space niches using immediate and delayed reinforcement. *Evolutionary Computing: AISB Workshop 1994*, 224–237.

Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. In Eiben, A., Bäck, T., Shoenauer, M., & Schwefel, H. (Eds.), *Parallel Problem Solving from Nature - PPSN IV* (pp. 178–187). Berlin: Springer Verlag.

Naudts, B., & Naudts, J. (1998). The effect of spin-flip symmetry on the performance of the simple GA. In Eiben, A. E., Bäck, T., Schoenauer, M., & Schwefel, H.-P. (Eds.), *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature PPSN V* (pp. 67–76). Berlin Heidelberg: Springer-Verlag.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1998). *Linkage problem, distribution estimation, and Bayesian networks* (IlliGAL Report No. 98013). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (2000). *Hierarchical problem solving by the Bayesian optimization algorithm* (IlliGAL Report No. 2000002). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., Goldberg, D. E., & Lobo, F. (1999). *A survey of optimization by building and using probabilistic models* (IlliGAL Report No. 99018). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In Roy, R., Furuhashi, T., & Chawdhry, P. K. (Eds.), *Advances in Soft Computing - Engineering Design and Manufacturing* (pp. 521–535). London: Springer-Verlag.

Schwarz, J., & Ocenasek, J. (1999). Experimental study: Hypergraph partitioning based on the simple and advanced algorithms BMDA and BOA. In *Proceedings of the Fifth International Conference on Soft Computing* (pp. 124–130). Brno, Czech Republic: PC-DIR.

Segen, J., & Sanderson, A. C. (1981). *Model inference and pattern discovery* (Technical Report CMU-RI-TR-82-2). Carnegie Mellon University.