Multimodal Optimization using Genetic Algorithms
An investigation of a new crowding variation and the relationship between parental similarity and the effect of crossover

Thomas Grüninger
grueni@mit.edu
Stuttgart University,

and

David Wallace
Assistant Professor, Department of Mechanical Engineering,
MIT Room 3-435, 77 Massachusetts Avenue,
drwallac@mit.edu
Massachusetts Institute of Technology I

Abstract

This paper investigates the extension of genetic algorithms to multimodal optimization (*niching methods*). After reviewing current niching methods, a new variation of a crowding technique, named the *struggle genetic algorithm*, is introduced. Replacement occurs only between similar individuals based upon a similarity measure if an offspring wins the competition. Using a suite of test problems, the performance of the *struggle genetic algorithm* and three other niching methods – *deterministic crowding*, *restricted tournament selection*, and *fitness sharing* - is empirically examined. For each test problem the struggle GA consistently located a more complete set of optimal solutions. The struggle GA has also performed well when compared to global methods (simple and steady state GA). Additionally, crossover's adaptive mechanism based upon the similarity measurement of parents is investigated for commonly used representations. Empirical investigations suggest that *real*-coded parameters are superior to the traditional *binary* or Gray-coding of continuous variables. For real-coded variables a newly designed crossover operator, the sphere-crossover, is introduced and tested. Preliminary test results illustrate its adaptive power.

1. Problem statement

Genetic algorithms (GA's) have been applied successfully to optimize several kinds of problems, where many traditional methods (such as the gradient search, the simplex method, etc.) are not applicable because of assumptions or requirements - such as the need for objective function derivatives, convexity, or the linear correlation of variables [Beasly2].
One general problem of optimization techniques is that there is not always a guarantee of convergence to he globally optimal solution. Frequently they converge to local optima. GA's may also have similar problems. One of the major problem is known as *premature convergence*, which means that all individuals in a population become nearly identical before the optima has been located. In the worst case, the GA also becomes trapped in a suboptimal solution, even though GA's are actually considered to be very robust [Goldb:GA].
In many optimization problems (i.e. design problems) a technique capable of locating multiple solutions is often desired because the final judgment, decision or selection between alternatives will be made by humans. A better understanding of the search space structure, again revealed by finding multiple solutions can be quite helpful. If the search space has multiple solutions, a traditional GA is only able to locate one of them, even though the multiple solutions might be of equal quality. Additionally, niching methods might overcome problems such as premature convergence and being trapped in local optima.
This paper explores the application of niching methods to multimodal domains, where a desire for locating multiple solutions exists.

2. Review of niching methods

Evolutionary strategies like genetic algorithms are a general optimization/search technique which imitate the principles of natural evolution and molecular genetics. Unlike other methods, genetic algorithms operate upon several solutions (a population). For an overview of genetic algorithms the reader is referred to [Beasly2,Da91,DeJong,Goldb:GA,Schoe]. Methods which allow GA's to allocate and maintain multiple different optimal / suboptimal solutions in a population are called niching methods or multimodal GA's. The inspiration for niching again stems from nature, where different species coexist through adaptation to different niches.

2.1. Niching methods

Crowding [DeJong] and *fitness sharing* [GuR] are the two primary approaches for preserving diversity and maintaining multiple solutions in a population.
A distance metric defined over the search space (which can be either *genotypic* or *phenotypic*) is used to distinguish the similarity of individuals in all niching methods. Crowding techniques use this measurement to replace favorably similar individuals, whereas fitness sharing uses the metric to derate an individuals' fitness by an amount according to the number of similar individuals in the population.

2.1.1. De Jong's crowding

De Jong [DeJong] introduced an algorithm which he called the *crowding factor model*. Specified by the generation gap G, $G \cdot PopSize$ individuals are chosen via fitness proportionate selection to create an equal number of offspring. For each of the offspring a random sample of CF (crowding factor) individuals is scanned from the current population. The offspring then replaces the most similar individual of this sample. De Jong used the Hamming-distance between two individuals as the measurement for similarity. Typical values for the crowding factor CF are 2 or 3 and 10% for the generation gap G. However, for the goal of maintaining multiple solutions in a population, it has been shown that De Jong's crowding is only of limited use [Mahf:Niching].

2.1.2. Deterministic crowding

S. Mahfoud [Mahf:Niching] proposed this variation of De Jong's crowding. Deterministic crowding first randomly groups all individuals in a population into parent pairs. Each pair generates two offspring by application of the genetic operators. Every offspring then competes against one of its parents. There are two possible parent-child tournaments, decided by the sum of the distances between the parents and the children of both possible combinations. The winner of the competitions move on to the next generation. The pseudocode for the two possible parent-child tournaments is given below:

```
IF [$d(P_1, C'_1) + d(P_2, C'_2) = d(P_1, C'_2) + d(P_2, C'_1)$]
        IF $f(C'_1)$ □ $f(P_1)$ replace $P_1$ with $C'_1$
        IF $f(C'_2)$ □ $f(P_2)$ replace $P_2$ with $C'_2$

ELSE
        IF $f(C'_1)$ □ $f(P_2)$ replace $P_2$ with $C'_1$
        IF $f(C'_2)$ □ $f(P_1)$ replace $P_1$ with $C'_2$
```

2.1.3. Restricted tournament selection

Another crowding method was proposed by Harik [Harik]. The parents are chosen randomly from the population. For each new child a random sample of CF (Harik refers to CF as w = *window size*) individuals from the population is formed. In contrast with De Jong's approach, the child then competes with the most similar individual in this sample. If the child is better, it replaces it.

2.1.4. Fitness sharing

Goldberg and Richardson [GuR] used Holland's [Ho75] sharing concept for niching. Every individual in a niche shares its fitness with all others in that niche. Niches with higher fitness values are able to support more individuals. In contrast with crowding techniques (where the replacement strategy influences diversity by replacing similar individuals), sharing uses fitness deration and the parental selection mechanism to affect population diversity. In addition to requiring a distance or similarity measure, a niche radius (threshold distance) is needed to define the niches for individuals.
The altered fitness value (shared fitness) of an individual is computed from its individual fitness divided by its niche count. The niche count of an individual is the sum of the sharing function values between itself and all individuals in the population (including itself). If two elements of a population are identical, the sharing function returns a value of 1. If two elements in a population exceed a certain threshold distance (also called niche radius σ_{share}) the sharing function returns a value of 0, implying that the individuals are in different niches and do not share their fitness values. A commonly used sharing function is defined as follows:

$$sh(d) = \begin{array}{ll} 1-(\frac{d}{\sigma_{share}})^{\alpha} & if\ d<\sigma_{share} \\ 0 & otherwise \end{array}$$

The parameter alpha controls the shape of the sharing function (the sharing function is linear for · =1). The derated fitness f^* of an individual i is given by:

$$f^*(i) = \frac{f(i)}{\sum_{j=1}^{PopSize} sh[d(i,j)]}$$

Along with the use of simple GAs (generational replacement) for sharing, overlapping populations may be used. In this scheme, new offspring are first added to the current population. Shared fitness values are computed and a number of individuals (equal to the number of children) are then eliminated from the population. This is accomplished by removing those individuals with the worst shared fitness values. For fitness proportionate selection to work properly on the next generation, the shared fitness values are then recalculated. This technique carries a higher computational overhead (distance comparisons) than fitness sharing with non-overlapping populations.

2.1.5. Sequential niching

This variation on fitness sharing was proposed by Beasley, Bull and Martin [Beasly]. Multiple solutions are found serially with sequential runs of a simple GA. The best solution of each run is stored. In order to prevent convergence to a previously located optima, the fitness values of solutions near previously found solutions (within a niche radius) are derated.

3 The struggle genetic algorithm

As reported in [Muehl], it is known that a finite population will always converge to a single genotype, assuming that the effects of mutation are negligible.This phenomenon, also called *genetic drift*, can be observed in genetic algorithms as well.
It is often observed in GAs that this point will not necessarily be an optimum (*premature convergence*). Without any mechanism to introduce new diversity, the GA converges on a single solution. When all individuals in a population have converged to a single genotype crossover does not contribute to the search since all the information it can process is equal. For this reason mutation is helpful. Mutation can on the

one hand help to explore[1] the search space (by big mutation steps) and also help to exploit[2] the search space (by small mutation steps). As the mutation rate increases more 'blind' diversity will be introduced, thereby helping to prevent the GA population from converging. However, relying on a truly random exploration such as mutation can be inefficient.
Thus, we studied the search role of crossover. When diversity is present crossover can help in both exploring and exploiting the search space if it adapts its scatter to the similarity of the parents (i.e. similar parents yield similar children, differing parents yield children with similar differences). For the crossover operator itself there is therefore no real difference in the mechanism of exploration and exploitation.
If we want to rely on crossover for exploration and exploitation (rather than high mutation rates), useful diversity is required. This desire leads to the question of how to preserve useful diversity in a population? Since the population has no knowledge of the search space, it should maintain all promising solutions. Otherwise the population might drift to the first or most easily found promising area and thus loose information that may lead to yet unencountered promising areas.Once lost, the chance that a mutation will regain that desired information is quite low. Therefore, it seems important to maintain as many as possible different promising solutions in the population.
The *struggle GA* was devised to have such characteristics. Diversity is preserved by minimizing the erroneous replacement of unique and potential promising solutions.As uniform selection is used to choose parents, the local competition between a newly generated child and the most similar individual in the population introduces selection pressure. The pseudocode for the *struggle GA* is given below:

```
REPEAT until stop criterion is met:
1.  Select parents $P_1$ and $P_2$ randomly (uniform selection)
2.  Cross parents $P_1$ and $P_2$, yielding an offspring $C$
3.  Apply mutation or other operators, yielding $C'$
4.  Find most similar individual $R$ to $C'$ in current population
5.  IF $f(C') \geq f(R)$ replace $R$ with $C'$
```

The struggle GA is a variation on crowding. It is functionally equivalent to (RTS) with a *CF* of *PopSize*. If a newly generated offspring wins the competition against its most similar individual in the *whole* population (sample size = *PopSize*) it replaces its opponent. Unlike fitness sharing (SH) and restricted tournament selection (RTS), no 'explicit' niches are defined (niche radius for SH) or assumed (RTS's window size *w* controls implicit the number of maintained niches). In contrast to deterministic crowding (DC), the whole population serves as the set for possible competitions, rather than only the parents.

4. Crossover and the similarity measurement

Genetic algorithms process similarities in the underlying encoding [Goldb:GA]. The representation (encoding) and crossover operator used are important for the successful application of a genetic algorithm.
Thus we questioned, it is important to investigate how crossover scatters offspring as a function of similarity to its parents ($d_{parents}$). Crossover operators capable of both exploration and exploitation are desired. We will first suggest meaningful phenotypic similarity measurements for a selection of problem types. Phenotypic similarity measurements are favored over genotypic, as genotypic distances may differ significantly from phenotypic distance.

4.1. Similarity measurements between individuals and the similarity average between parents

4.1.1. Hamming-distance for bitstring variables

[1] To locate convexity regions
[2] to optimize within local convexity regions

The Hamming-distance is an appropriate phenotypic measurement of similarity for problems where a bitstring representation matches the phenotypic meaning (i.e, ONEMAX [Sys]). The Hamming-distance may also be appropriate for bitstring representations where no *a priori* knowledge of the problem can be used to define a phenotypic distance metric. The Hamming-distance between two bitstrings s_1 and s_2 of length l is given by the sum of the non-matching bit-values at each position. In general, the Hamming-distance d_h between two bitstrings can be written as:

$$d_h(s_1, s_2) = \sum_{i=1}^{l} \left| v\left[s_1(i)\right] - v\left[s_2(i)\right] \right|$$

where $v[s_j(i)]$ is the bit-value of string j at the position i. The similarity average $m_p(i)$ of two parental bitstrings is determined by the bitwise average of the parental bitstrings. Note that the sum of the values of $m_p(i)$ equals the average number of number 1's in the parental strings (equals also l minus the average number of 0's). The equation for m_p is given below:

$$m_p(i) = \frac{v[s_1(i)] + v[s_2(i)]}{2}$$

4.1.2. Euclidean distance for real variables

In an n-dimensional search space where all independent variables x_i ($i = 1, \dots n$) are continuous, the Euclidean distance between two points $\vec{x}_1$ and $\vec{x}_2$ is defined as:

$$d_e(\vec{x}_1, \vec{x}_2) = \sqrt{\sum_{i=1}^{n} \left(x_{1i} - x_{2i}\right)^2}$$

This distance measurement seems appropriate as a similarity function provided the magnitude of the variables x_i are similar. However, scaling may be an issue in many problems, when the difference in scale might reach several orders of magnitude. In this case, a normalized Euclidean distance seems to be more appropriate. For n continuous variables x_i ($i = 1, \dots n$), with the lower bound b_{li} and the upper bound b_{ui} for x_i ($x_i \in [b_{li}; b_{ui}]$), the normalized distance can be written as:

$$d_e^0(\vec{x}_1, \vec{x}_2) = \sqrt{\sum_{i=1}^{n} \left(\frac{x_{1i} - x_{2i}}{b_{ui} - b_{li}}\right)^2}$$

The similarity averages for both cases can then be expressed as a vector average of the parental vectors. This is described below:

$$m_p = \frac{\vec{x}_1 + \vec{x}_2}{2} \quad \text{and} \quad m_p^0 = \frac{\vec{x}_1^{\,0} + \vec{x}_2^{\,0}}{2}$$
.

4.1.3. Common-edge distance for order-based problems

For an order-based problem like the traveling salesman problem (TSP), we suggest an appropriate distance measurement accounts for the number of common edges between two tours Π_1 and Π_2. E_1 and E_2 are the connection-matrices of the tours Π_1 and Π_2. The elements (i, j) of the matrices contain a value of 0 if a connection between the cities t_i and t_j does not exist and a value of 1 if the connection exists within the given tour. The distance d_{ce} for two tours with n cities is:

$$d_{ce}(\Pi_{1,}\Pi_2) = n - \sum_{i=1}^{n}\sum_{j=1}^{n}[E_1(i,j) \text{ and } E_2(i,j)]$$

This similarity measurement (or some variation thereof) could also be used for other sequence-based problems. How to calculate the similarity average between two parent tours is not obvious. This problem is addressed in section 4.2.2.

4.2. Investigation of crossover scatter around the similarity average

An empirical investigation of how different crossover operators scatter offspring in relation to parental similarity was performed. Parents with distinct distances ($d_{parents}$) were generated randomly and crossed. Then the offspring distance from the parental *similarity average* (m_p) was calculated. These data were accumulated to obtain statistical scatter-distributions. Two types of analyses, σ-scatter and distribution-scatter plots, were performed.

σ-scatter plots:
n pairs of parents P_1 and P_2 were generated randomly for each possible distance $d_{parents}=d(P_1,P_2)$ between parents. An offspring C was generated for every pair of parents by crossing P_1 and P_2. Then the distance $d_{average}$ from an offspring to the similarity average m_p was calculated. By calculating and plotting the standard deviation σ of the n generated offspring C from the similarity average for each distance $d_{parents}$ of parents, the σ-scatter plots were obtained ($d_{average}$ was negated for offspring closer to P_1 than to P_2).

Distribution-scatter plots:
The same calculations as for the σ-scatter plots were performed, but instead of looking at standard deviation, histograms for each distance $d_{parents}$ were constructed. This will show us how the crossover operator scatters around the parent's similarity average.

Crossover for bitstring representations of bitstring variable problems
The uniform, one-point and two-point crossover operators are considered for bitstring representations using the Hamming-distance as the similarity measurement. One classical example where the Hamming-distance matches the phenotypic meaning is the ONEMAX problem (also known as the counting 1's problem). In this domain, the fitness of an individual equals the number of 1's in the bitstring. For our investigations a bitstring of length l=100 and a sample size of n=2,000 was used. The distribution-scatter plots are shown in figure 1. The σ-scatter plots are not shown for this case, but may be found in [mythesis].

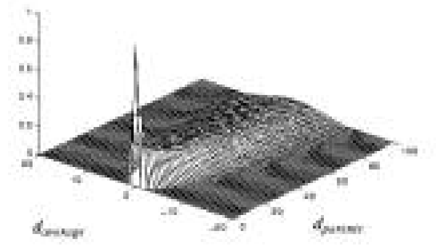


Figure 1. Distribution-scatter plots for bitstring representations with the Hamming distance as the similarity measurement.

The distributions in figure 1 can be explained mathematically. For uniform crossover the scatter forms a binomial distribution because for each distance $d_{parents}$ there is always a fixed number n_d=$d_{parents}$ of differing bits in the parents. For each differing bit-value at a position in the parental strings, uniform crossover transfers 1's and 0's with equal probabilities to the child (p=q=0.5) so that the standard deviation equals :

$\sigma = \sqrt{n_d pq} = \frac{\sqrt{d_{parents}}}{2}$.

One-point crossover uses one substring from each parent to generate an offspring, so the percentage of differing bits in the child relative to its parents is distributed uniformly with a standard deviation of:

$\sigma = \frac{d_{parents}}{\sqrt{12}}$.

The difference between one-point crossover and two-point crossover vanishes with an increasing length of the bitstrings.

The investigations also show that for this case uniform, one-point, and two-point crossover adapt their scatter to the similarity of the parents ($d_{parents}$). Interestingly, Syswerda [Sys] showed that for the ONEMAX problem *uniform crossover* works better. This may be significant. The ERX-operator investigated in section [sec:orders} also shows a similar binomial-like distribution and ERX outperforms the PMX-operator (which has a uniform-like distribution).

4.2.2. Crossover for order-based problems

In this section *partially-matched crossover* (PMX) [Goldb:GA] and *edge-recombination crossover* (ERX) [Whitley] are investigated for order-based representations such as used for the traveling salesman problem. PMX works by first choosing two crossing sites randomly and exchanging the strings in between. Then it 'repairs' the offspring as the exchange of subtours may produce invalid tours. ERX's operation is quite different. It tries to incorporate as many as possible edges from both parents into the offspring.

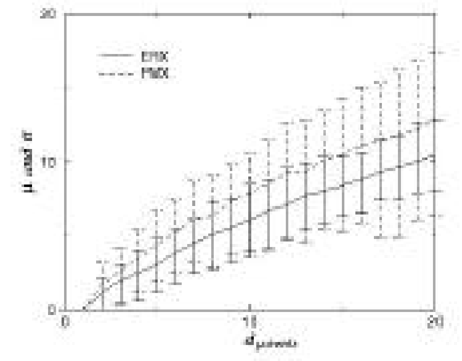
As mentioned earlier is not obvious how to calculate the similarity average m_p for two parent tours. Thus we will draw upon analogies with crossover for bitstrings using the Hamming-distance as the similarity measurement.

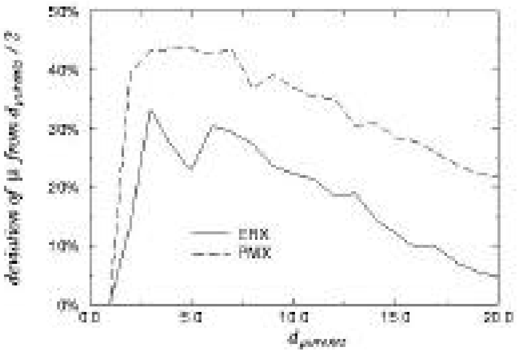
Uniform crossover for bit strings can be regarded as a process which transfers as many as possible common bit-values to the offspring, since equal bit-values at a position in the parental strings will be incorporated into the child (quite similar to ERX's transfer mechanism for common edges). PMX uses a similar principle as two-point crossover for bit strings. Both operators incorporate a substring from one parent and the remaining information from the second parent into the child. For the bitstring case, the scatter distribution around the similarity average is the same as if one calculates the scatter distribution around the mean $\mu = \frac{d_h(C,P_1) + d_h(C,P_2)}{2}$ for $d_1{=}d_h(C,P_1)$ and $d_2{=}d_h(C,P_2)$. The sum $d_1{+}d_2$ always equals $d_{parents}$ and the scatter is symmetrical around the mean $\mu = \frac{d_1 + d_2}{2} = \frac{d_{parents}}{2}$.

As an approximation we assumed that this is also true for the order-based representations.The assumption seemed reasonable because an offspring is generated primarily by incorporating edges from both parents (a transfer of subtours is also a transfer of edges). By considering the common edges of a child relative to its parents, a child should have on average half its edges from each parent. Thus, μ should be approximately the same as $\frac{d_{parents}}{2}$ .The empirical results (shown in figure [fig:orddev:b]) support our assumptions. Therefore, for every possible distance $d_{parents}$ the mean distance μ and the standard deviation σ of the n generated offspring C relative to their parents P_1 and P_2 were calculated using equations 9.

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\frac{d_{1i} + d_{2i}}{2} \text{ and } \sigma = \sqrt{\frac{1}{2n}\sum_{i=1}^{n}(\mu - d_{1i})^2 + (\mu - d_{2i})^2} \qquad (9)$$

By plotting the mean μ and the standard deviation σ for each distance $d_{parents}$, μ-σ scatter plots were obtained for the order-based representation and the crossover operators ERX and PMX (see figure 2.a). A sample size of n=200 was used. Note that a distance between two parents of $d_{parents}$,=2 can never occur because the smallest change in a tour always effects two edges.

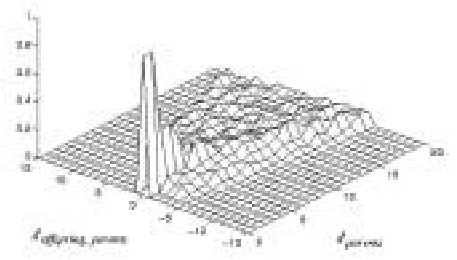


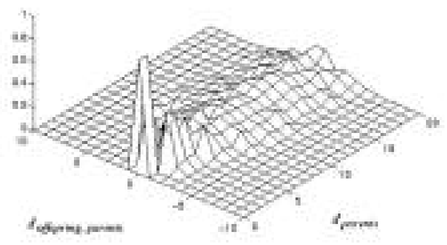


(a) μ-σ -scatter plots for PMX and ERX **(b)** Deviation of μ from $\frac{d_{parents}}{2}$.

Figure 2. Investigation of PMX and ERX for order-based representations with the common-edge distance as the similarity measurement.

For the distribution-scatter plots in figure 3 the calculated histograms for each distance $d_{parents}$ were shifted by their mean μ.





(a) PMX **(b)** ERX

Figure 3. Distribution-scatter plots for (a) PMX and (b) ERX with the common-edge distance as the similarity measurement.

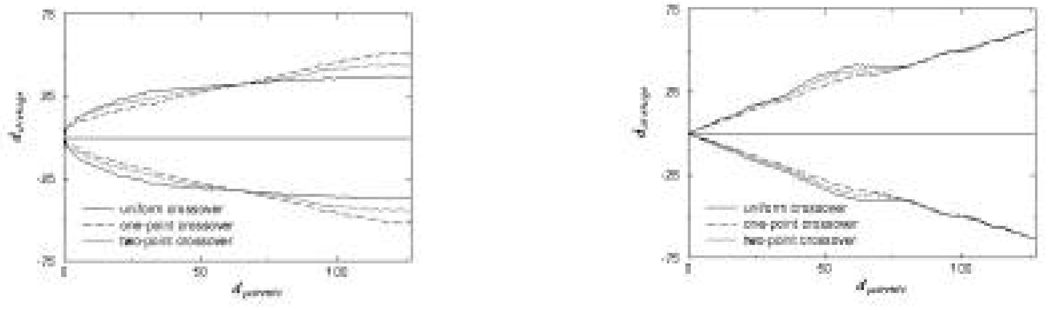
Both operators adapt their scatter to the similarity of parents. Interestingly, the plots in figure 3 illustrate that PMX shows roughly a uniform distribution and ERX a binomial-like distribution around the mean μ. As observed for bitstrings, the binomial-like scatter distribution of the ERX-operator works better for the TSP than the uniform of PMX [Schoe].

4.2.3. Crossover for binary and Gray-code representations of real variable problems

Real-variable problems are commonly represented by encoding each variable as a bitstring. To convert a real number into a bitstring there are two methods. The first uses standard *binary*-encoding and the second *Gray*-encoding. We will investigate the operators *uniform*, *one-point* and *two-point crossover* for the real number interval [0; 2^7-1 = 127] mapped to a bitstring of length l =7. Since we are interested in the *phenotypic* similarity we will use the Euclidean distance as the similarity measurement. The similarity average m_p of two parents was calculated by averaging the decoded values of the parental bitstring. We performed n=5,000 crossovers, yielding n=5,000 offspring for each distance $d_{parents}$. The σ-scatter plots are shown in figure 4. Representative distribution-scatter plots for two cases are shown in figure 5. For a complete set of plots the reader is referred to [mythesis].

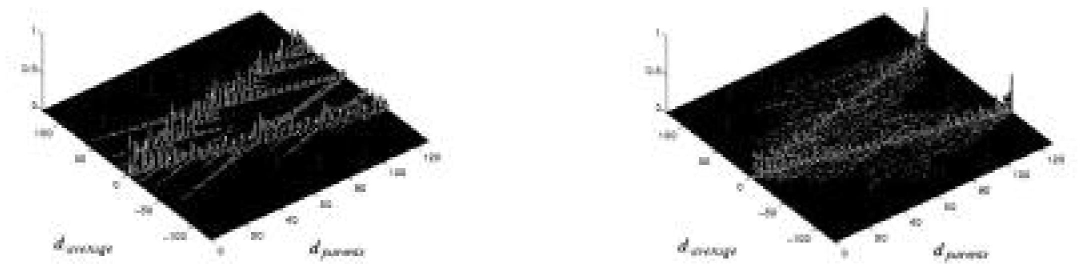
In all 6 cases investigated, the crossover operators only sampled distinct points around the similarity average. As expected, uniform crossover scattered more evenly than one or two-point crossover. However, in all cases several regions were never sampled! The scatter-distributions are inconsistent (for different parental distances) and discontinuous.

All encodings (together with the crossover operators) raised concerns about the ability to adapt between exploration and exploitation as a function of parental similarity. For binary coding (figure 4.a), scatter is relatively wide for small parental distances.



(a) *binary*-coded **(b)** *Gray*-coded

Figure 4. σ-scatter plots for *binary* and *Gray*-code representations with the Euclidean distance as the similarity measurement.



(a) *one-point crossover* with *binary*-coding **(b)** *uniform crossover* with *Gray*-coding

Figure 5. Distribution-scatter plots for (a) *binary* and (b) *Gray*-code representations with the Euclidean distance as the similarity measurement.

With Gray-coding (figure 4b) the likelihood of producing an offspring between parents drops to zero as the distance between parents increases. However, the offspring of similar parents with smaller distances do not get scattered as widely about the similarity average as for the binary-coded case. This indicates that Gray-coding has a better ability to exploit a region between similar parents and may support Hollstien's claim that Gray-coding tends to work better than binary encoding [Goldb:GA].

In addition to the results presented here, other analyses were performed. In summary, the characteristics of each scatter-distribution are independent of the number of bits used to encode a real number. Only the granularity decreases. As the bitstring length *l* increased, the distributions showed a fractal character (repeating structures). As more bits were used, the patterns repeated and became 'deeper'. For a 3-allelic representation the results were similar to the reported 2-allelic case, only the patterns changed. For a 4-allelic representation, the results were similar to the 2-allelic case presented here.

4.2.4. Crossover for representations of real variable problems: Development of the sphere-crossover operator

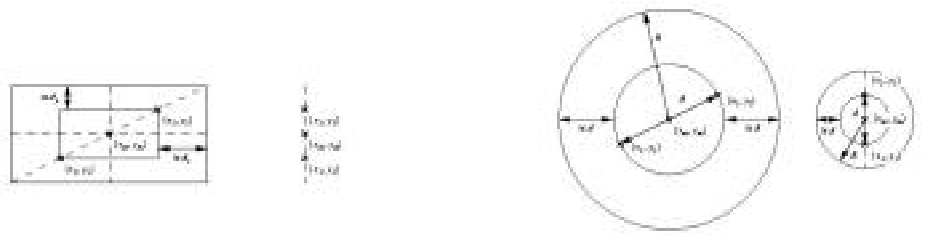
Practical experience has shown that real-coded genes usually work better than binary or Gray-coded genes for functions of continuous variables [Da91]. Interestingly enough, the *Evolutionsstrategien* (evolutionary strategies) introduced by Rechenberg in the early seventies [Rech] used real-coded genes from the onset. Several other reasons speak for the use of real number representations [Eshel], such as eliminating the need to specify an adequate precision for the encoding.

Empirical tests on a variety of test functions, indicate that a crossover operator named *blend*-crossover (BLX-α), especially the version BLX-0.5, is a powerful operator [Eshel].

This operator works by calculating the average value of the corresponding genes in parents and applying a uniform scatter around the average. The interval for the uniform scatter is distinguished by the parameter α. In figure 6 the BLX scheme is provided for a two-dimensional real-coded chromosome. The parents selected for mating (P_1 and P_2) have the chromosomes (x_1,y_1) and (x_2,y_2) with the corresponding distances $d_x = |x_2 - x_1|$ and $d_y = |y_2 - y_1|$ for each dimension. The average values for both dimensions are
$x_m = \frac{x_1 + x_2}{2}$ and $y_m = \frac{y_1 + y_2}{2}$.
However, as displayed in figure 6, BLX-α's scatter depends on the distances d_{xi} for each dimension *i*, and therefore the shape of the scatter area varies for parents of a distinct Euclidean distance. For example, if two corresponding gene values in the parents are equal, the child's gene will be the same as in parents even though there may be a tremendous difference in the parent's genes in all other dimensions. Is it reasonable to exploit in some dimensions when other dimensions still require exploration? Probably not. In addition, BLX-α does not scatter consistently depending on the Euclidean distance of parents (varying scatter areas for parents with the same distance). Also, the probability that points are sampled close to the similarity average is low (see distribution-scatter plot in figure 7).

Therefore, we developed a new Euclidean distance-based crossover operator for real number representations called *sphere-crossover* (SPHX-α). It generates the gene value for a child by first calculating the average vector $\vec{x}_m$ of the parents (the scheme is illustrated in figure 6). Then, a point within the hypersphere around $\vec{x}_m$, defined by a distribution type and the parameter α, is chosen. As SPHX-α operates directly on the Euclidean distance, its adaptive power may be higher than BLX-α's.

However, there are several possible distributions for SPHX's scatter. In accordance with observations made in the previous sections, testing a uniform and normal distribution of the child's distance from the similarity average (the parental vector average) seemed appropriate.



(a) BLX-α **(b)** SPHX-α

Figure 6. Schemes of (a) *blend-crossover* and (b) *sphere-crossover* for a two-dimensional real-coded representation.

As reported in [Eshel], only BLX-0.5 has the property, that in absence of selection pressure, the probability of an offspring lying outside the range of its parents is the same as the probability that a child will lie between its parents. Therefore we used α=0.5 for sphere-crossover with a uniform distribution (SPHX-0.5). For sphere-crossover with a normal distribution of the child's distance to the parental vector average (SPHX-Gauss) we used $\alpha = \infty$ and $\sigma = 0.74d_{parents}$, so that the convergent and divergent tendencies are balanced.

In figure 7, the scatter-distributions of *a*) BLX-0.5, *b*) SPHX-0.5, *c*) SPHX-Gauss for a distance $d_{parents}$ of P_1 and P_2 are shown (for a two-dimensional case). A sample size of n = 100,000 was used. Since BLX and SPHX scatter offspring with a consistent distribution form for all parental distances, only one histogram for a distinct distance $d_{parents}$ was calculated (both adapt their spread to the parental distance). However, there are other possible distributions for SPHX, such as i.e. *d*) a uniform distribution of the child's coordinates within the hypersphere ($\alpha = \frac{\sqrt{2}}{2}$), which have not been studied so far. The scatter-distribution for case *d*) is also shown in figure 7.

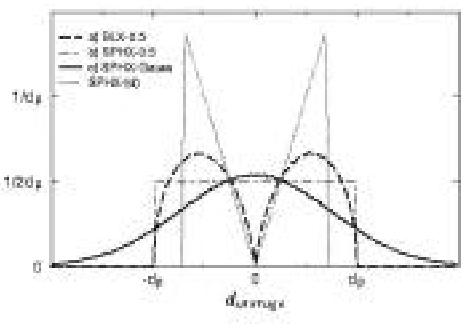


Figure 7. Distribution-scatter-plot for *a*) BLX-0.5, *b*) SPHX-0.5, and *c*) SPHX-Gauss with a Euclidean distance of d_p between the parents. Additionally, SPHX's distribution-scatter plot for *d*) a uniform distribution of the child's coordinates within the hypersphere is shown.

A performance comparison for BLX-0.5, SPHX-0.5, and SPHX-Gauss will be provided in the following section.

5. Empirical evaluation of the struggle GA with global GAs for global real-value function optimization

The performance of the struggle GA will first be compared to global GAs (steady state and simple). We will also compare the performance of BLX-0.5 with the new SPHX-0.5 and SPHX-Gauss (see section 4.2.4). The struggle GA is compared to a steady state GA. We also applied a simple GA but on average it performed worse than the steady state GA on the test problems.

5.1. Test problems

Comparisons will be presented for the following two numerical minimization problems:

Unimodal function:

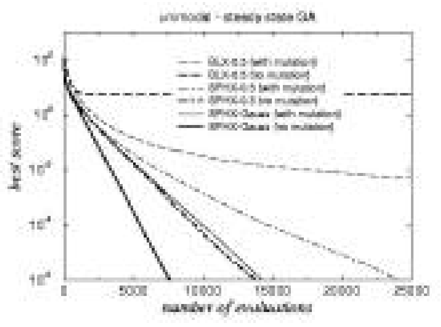
$$f\left(x_i \,|_{i\,=\,1,\cdots n}\right) = \sum_{i=1}^{20} x_i^2 + \left(\sum_{i=1}^{20} x_i\right)^2, \qquad x_i \quad [-5;5]$$

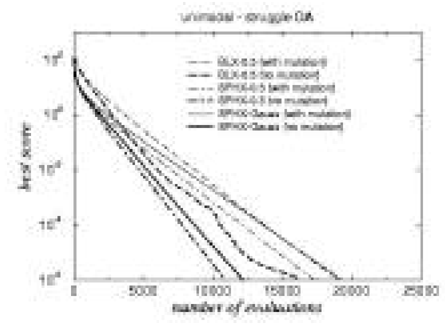
Griewank's function:

$$f\left(x_i \,|_{i\,=\,1,\cdots n}\right) = \sum_{i=1}^{20} \frac{x_i^2}{4000} - \prod_{i=1}^{20} \cos\left(\frac{x_i^2}{\sqrt{i}}\right), \qquad x_i \quad [-600;600]$$

Test results

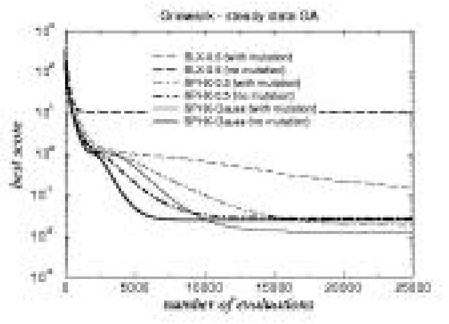
The global minimum for both functions is located at $\vec{x}^* = \vec{0}$ with the optimal score of $f(\vec{x}^*) = \vec{0}$ A steady state GA (with *G*=20%) and a struggle GA (with the Euclidean distance as the similarity measurement) were used for the simulations. For both algorithms a population size of 25 individuals was used. In order to see better the merits of the different crossover operators, full crossover (p_c=100%) was applied and runs with mutation (p_m=2.5%) and without mutation (p_m=0%) were conducted. Our mutation operator resets each variable with a probability of 2.5% with a random value. This mutation will tend to introduce new diversity into the population but will not tend to help individuals to exploit a region. Figures 8-9 display results averaged over 100 runs (always using the same set of 100 different random seeds). Every run was stopped after 25,000 function evaluations.



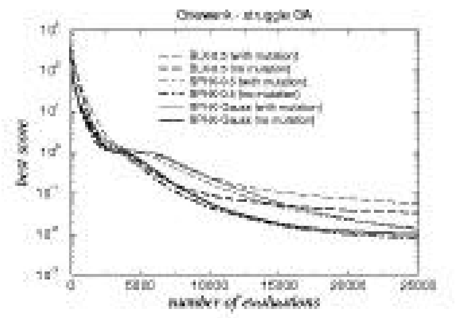


(a) *steady state* GA **(b)** *struggle* GA

Figure 8. Simulation results for the *unimodal function*. Best score: $f(\vec{x}^*) = \vec{0}$



(a) *steady state* GA



(b) *struggle* GA

Figure 9. Simulation results for *Griewank's function*. Best score: $f(\vec{x}^*) = \vec{0}$

For all the crossover operators used in the study, on average the struggle GA more reliably approached the global optimum than the steady state GA. The performance of the steady state GA was highly dependent upon the crossover operator used. Another interesting observation is that for the steady state GA the mutation operator (which scatters widely to introduce only 'blind' diversity) helped BLX, whereas the mutation operator rather hindered SPHX in most cases. This may be explained by BLX's characteristic that allows it to converge in one dimension while others are still quite different. For the struggle GA, where a useful diversity is maintained, our random mutation operator did not help either BLX nor SPHX. Additionally, for all algorithms and problems used, both SPHX-0.5 and SPHX-Gauss outperformed BLX-0.5 as predicted. For the steady state GA, SPHX-Gauss tended to work better than SPHX-0.5, whereas for the struggle GA SPHX-Gauss performed slightly worse than SPHX-0.5.

6. Empirical comparison of the struggle GA with other niching methods}

The performance of *deterministic crowding* (DC), *fitness sharing* (SH), *restricted tournament selection* (RTS) and the proposed *struggle GA* (STR) were compared. Test problems with varying characteristics were used to observe the behavior of the different niching methods.

6.1. Test problems

6.1.1. Shekel's foxholes

This two-dimensional test function has 25 equally spaced peaks of varying height. The original function from De Jong's dissertation [DeJong] was transformed into a maximization problem. The equation of this function is given below:

$$z_1(x,y) = 100 - \frac{1}{\frac{1}{50} + \sum_{i=1}^{25} \frac{1}{1+i+(x-a_i)^6+(y-b_i)^6}}$$

with $a_i = 16[(i \mod 5) - 2]$ and $b_i = 16[(i \div 5) - 2]$.

6.1.2. Hilly function

Frequently, a wide variety of one-dimensional sinusoidal test functions with different properties are used to test niching methods. Here, we designed a two-dimensional function with unequally-spaced peaks of non-uniform height. In the x-direction the height decreases much more than in the y-direction (from the origin). Additionally, a global optima is added far from the generally promising area near the origin. For $x,y \in$ [-100;100], this function (as defined in equation 13 and shown in figure 10) has 36 peaks and its global optima is located at (b, b).

$$z_2(x,y) = 10\left[e^{-\frac{|x|}{50}}\left(1-\cos\left(\frac{6}{100^{\frac{3}{4}}}\pi|x|^{\frac{3}{4}}\right)\right) + e^{-\frac{|y|}{250}}\left(1-\cos\left(\frac{6}{100^{\frac{3}{4}}}\pi|y|^{\frac{3}{4}}\right)\right) + 2\left(e^{-\frac{(b-x)^2+(b-y)^2}{50}}\right)\right],$$

with $b = \left(\frac{5}{6}\, 100^{\frac{3}{4}}\right)^{\frac{4}{3}}$.

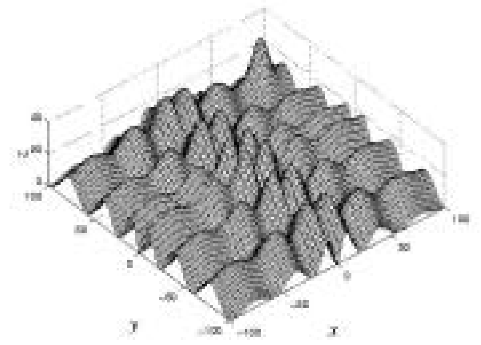


Figure 10. Graph of the *hilly function*

6.1.3. Order-five deceptive problem

Another class of test problem was an order-five deceptive problem [GuD] with ten concatenated fully deceptive five-bit subfunctions, altogether forming a 50-bit problem. Each subfunction has two complementary attractors (a deceptive one at 00000} and a global one at 11111. Subfunction and function values related to the number of 1's (unitation) are shown in figure 11. This 50-bit problem has a total of $2^{10\,=}$ 1024 optima, of which only one is the global. Goldberg investigated this problem with a fast messy GA [GuD] , where in the evaluation only the global optima was considered. Likewise we will investigate the ability of niching methods to locate the global maxima.

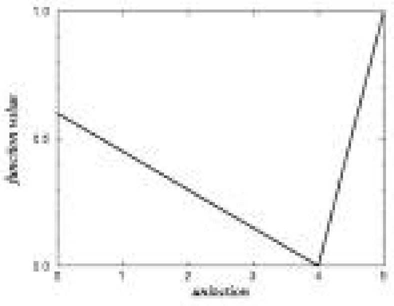


Figure 11. Graph of the deceptive five-bit subfunction.

6.1.4. Grid-TSP problem

The traveling salesman problem (TSP) is representative for a class of combinatorial optimization problems. Given *n* cities the task for the salesman is to visit all cities only once so that the overall length of the tour is minimal. In order to build a problem with multiple global optima, twenty cities were arranged on an evenly spaced rectangular (5x4) grid. Since we chose the grid-distance to be 1.0 so the 14 global optima (shown in figure 12) have all tour lengths of 20.0.

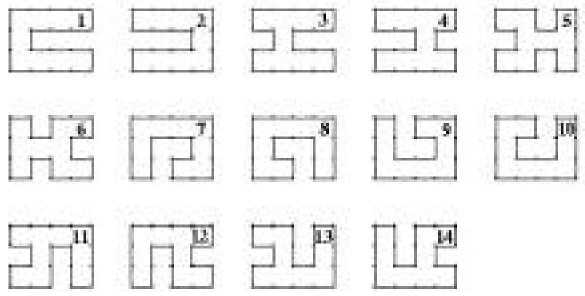


Figure 12. Labeled global optima for the *grid*-TSP problem.

6.2. Test results

All results shown were obtained with full crossover (p_c=100%) and a mutation rate p_m of 0%. We did not use mutation so as to better observe the characteristic behavior of each algorithm. A population size of 100 individuals was used for each test problem. For all averaged data the same ten random seeds were used. Every run was terminated after 200 generations. A new generation was defined as when PopSize new individuals were created and evaluated, effectively comparing the number of fitness evaluations.
Table 1 shows the settings used for the different problems. Values marked with * are from Mahfoud's Ph.D. dissertation [Mahf:Niching]. When using (RTS) the *window size w* was usually set to half of the population size. Additionally, sometimes some extra runs were performed with different parameters (also shown in the table). For the order-five deceptive problem the Hamming-distance was used as the similarity measurement (simulates a distance measure that utilizes no problem-specific knowledge).

Test problem	*similarity measurement*	*Crossover*	σ_{share} for (SH)	α for (SH)	*w* for (RTS)
Shekel's foxholes	Euclidean	BLX-0.5	8.0$^{*)}$	2.0$^{*)}$	50
hilly function	Euclidean	BLX-0.5	10.0	1.0	50
order 5-deceptive	Hamming	two-point	1.3 and 5	1.0	50

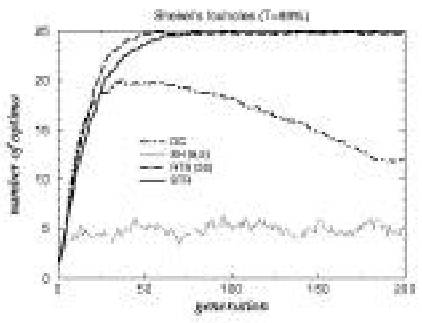
grid-TSP	common-edge	ERX	2.0	1.0	25 and 56

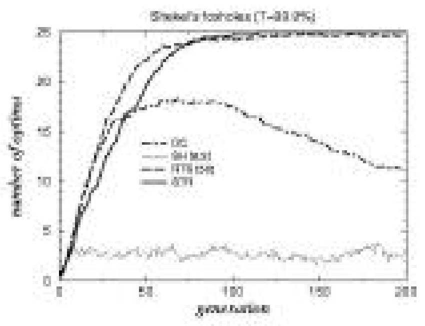
Table 1. Parameters and settings for the test problems

In all cases, fitness sharing with a generational replacement (as described in section 2.1.4) was not able to solve the test problems, so we used the modified version of fitness sharing (with overlapping populations, also described in section 2.1.4) which behaved slightly better.

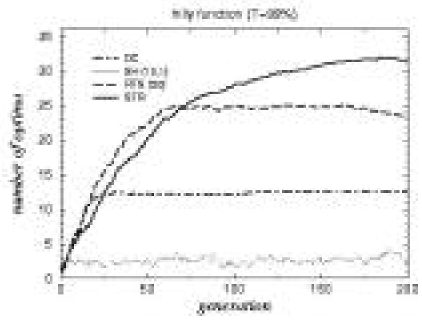
Two different thresholds (T_1 = 99% and T_2 = 99.9%) were used for the two-dimensional functions to determine whether an individual is considered to be at an optimal point. An individual was considered be at an optimum if its fitness exceeded the threshold percentage of the maximum fitness at that peak.

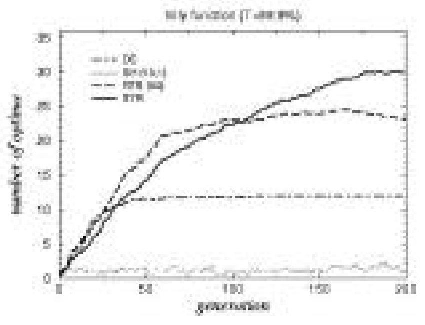
In figures 13-16 the test results are displayed.





(a) T_1=99%
(b) T_2=99.9\%
Figure 13. Maintained optima *versus* generation for *Shekel's foxholes*.

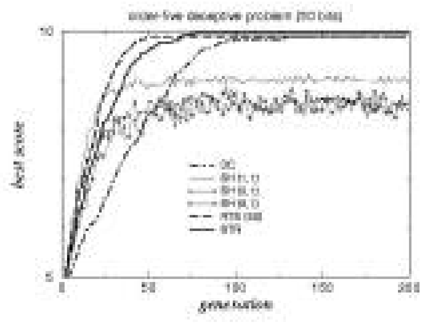




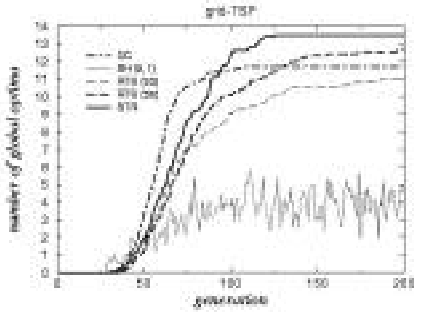
(a) T_1=99%
(b) T_2=99.9\%
Figure 14. Maintained optima *versus* generation for *hilly function*.

	(DC)	(SH)	(RTS)	(STR)
Thresold T_1 = 99%	90%	6%	88%	98%
Thresold T_2 = 99%	80%	0%	84%	90%

Figure 15. Average percentage of locating an maintaining the 5 best solutions for the *hilly function* (the global optimum and the four best suboptima).



(a) ***order-five deceptive*** **problem**



(b) ***grid*****-TSP problem**

Figure 16. (a) Best solution *versus* generation and (b) Maintained global optima *versus* generation.

6.3. Discussion

Although the suite of test problems does not serve as a basis for wide generalization, it is quite astonishing how a given niching algorithm behaved in the same way over the range of test problems. (DC) does not reliably yield all optimal solutions. This can be explained by considering the mechanism of replacement. Since the effect of crossover depends on the crossover type itself and the problem-specific landscape, the assumption that offspring are similar to their parents is not always true. Thus, individuals can leave peaks in favor of others (due to replacement errors), even though they might be not better. Therefore, (DC) often looses useful diversity. This limits exploration and exploitation of the search space. One advantage of (DC) is that there is no need for problem-specific parameters and the computational requirements to compare individuals is only of order $O(n)$.

In case of (SH), the implicit assumption about evenly spaced peaks (due to niche radius) seems to result consistently in poor performance (demonstrated by the hilly function results). Even when the optima are evenly spaced, optimal solutions can be lost due to (SH)'s fitness deration. For this reason, many examples in the literature scale fitness values (e.g., exponentially) so that this problem is minimized. The behavior of (SH) might be described more as a distributional effect. Consider a state during the evolutionary process where (SH) has found a near-to-optimal solution. As soon as a second solution enters the influence area of this individual (the niche), the fitness of both individuals will be derated, thereby increasing the chances that they will disappear. A very large population size is typically used in order to reduce this behavior. In addition, (SH) requires distance calculations of at least order $O(n^2)$. During our investigations, we observed that attempting to maintain many individuals fitness value peaks may hinder the search because useful diversity is abandoned to maintain multiple solutions at each peak. Thus, the number of located optima is lower and the more the population tends to drift towards a limited set of local solutions. In case of (STR), where the replacement errors are minimized, the chance that the algorithm is misled is reduced and more optima are found. The investigations reveal no obvious reason to use (RTS), which requires a window size, in favor of (STR). While (STR) increases the order of distance calculations to $O(n^2)$ compared to an order somewhere between $O(n)$ and $O(n^2)$ for (RTS) (the order depends on the window size w), the improved effectiveness of (STR) seems to justify some additional computation.

7. Guidelines for using the struggle GA

Experiences using the struggle GA suggest that it is desirable to use a similarity measure with a meaning that matches the similarity in the phenotypic search space. Consider the binary encoding of real values. Two nearby real values could have a quite big genotypic Hamming-distance, whereas two real values far away from each other could have an almost negligible genotypic Hamming-distance. This is illustrated below:

binary strings s_1 and s_2	*Hamming distance d_{max}* = 5	*Euclidean distance d_{max}* = 31
10000, 01111	5	1
10000, 00000	1	16

By using the Hamming-distance as the similarity measurement in this case, many replacement errors will occur and therefore a niching algorithm will perform poorly. Additionally, the crossover operator should adapt its scatter according to the phenotypic similarity of the mated parents. Even though it is not clear which distribution form is the best for a particular problem, a discontinuous and inconsistent form (for parents of distinct distances) appears to be disadvantageous. Interestingly enough, this observation about crossover seems to strongly affect traditional genetic algorithms (simple and steady state). Domain knowledge should be incorporated into a meaningful distance metric and the design of a crossover operator, which has the identified properties of adapting its scatter to the parents similarity.

8. Summary

In this paper we have investigated and tested niching methods for both global optimization and the location of multiple solutions in several domains. A new crowding variation, the struggle GA, was proposed for multimodal optimization. Newly generated offspring must be better than the most similar individual in a population in order to survive. This mechanism minimizes replacement errors, thereby reducing the algorithm's chance of being misled. Additionally, no problem-specific parameters for the algorithm are required. Empirical test results show that, on average, a more complete set of optimal solutions is located and maintained compared to other investigated niching methods. Further improvements might be achieved if parents were also selected based upon similarity. This is a subject for future work.

A close relationship between crossover's scatter and the similarity of parents was shown. Every crossover operator examined scattered with an operator-specific distribution around the similarity average of the parents and adapted its sampling area according to the distance (similarity) of the parents. This adaptive mechanism of crossover seems to support its unique capability of exploring and exploiting a search space. A binomial-like and normal distributions of the offspring from the parental similarity average seems to be advantageous. The traditional crossover operators for binary and Gray-coded continuous variables exhibit characteristics quite different to the other investigated cases. The scatter distributions of these operators are inconsistent and discontinuous. These anomalies may contribute to the observed superiority of real-coded chromosomes over the binary and Gray-coding of real-value parameters.

An understanding of crossover's properties can be used to design new operators. Building on observations of crossover operators (in relation to the similarity measure) a new crossover operator - the sphere-crossover - was designed for real-coded variables. It adapts the scatter of children according to the Euclidean distance of the parents. Test results demonstrate its capabilities relative to BLX.

References

[Beasly] D. Beasley, D. R. Bull, and R. R. Martin, *A Sequential Niche Technique for Multimodal Function Optimization*, Evolutionary Computing, 1(2), pages 101--125, 1993

[Beasly2] D. Beasley, D. R. Bull, and R. R. Martin, *An Overview of genetic Algorithms: Part 1, Fundamentals*, University Computing, 15(2), pages 58--69, 1993

[Da91] Lawrence Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991

[DeJong] Kenneth A. {De Jong}, *An analysis of the behavior of a class of genetic adaptive systems*, University of Michigan, Ann Arbor, 1995, Dissertation Abstracts International 36(10), 5140B; UMI 76-9381

[Eshel] L. J. Eshelman and J. D. Schaffer, *Real-Coded Genetic Algorithms and Interval-Schemata*, In L. Darrel Whitley (ed.), Foundations of Genetic Algorithms 2, pages 187--202, Morgan Kaufmann Publishers, San Mateo, CA, 1992

[Goldb:GA] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, 1989

[GuR] David E. Goldberg and J. Richardson, *Genetic Algorithms with sharing for multimodal function optimization*, In J. J. Grefenstette, Proceedings of the Second International Conference on Genetic Algorithms, pages 41--49, Lawrence Erlbaum Associates, Hillsdale, NJ, 1987

[GuD] David E. Goldberg, K. Deb, H. Kargupta, and G. Harik, *Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms*, In J. D. Schaffer (ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, pages 56--64, Morgan Kaufmann Publishers, 1993

[mythesis] Thomas Grfininger, *Multimodal Optimization using Genetic Algorithms*, Master Thesis, Stuttgart University

[Harik] Georges Harik, *Finding Multimodal Solutions Using Restricted Tournament Selection*, In J. D. Schaffer (ed.), Proceedings of the Sixth International Conference on Genetic Algorithms, pages 24--31, Morgan Kaufmann Publishers, 1995

[Ho75] J. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975

[Mahf:Niching] Samir W. Mahfoud, *Niching Methods for Genetic Algorithms*, University of Illinois at Urbana-Champaign, 1995, IlliGAL Report 95001

[Muehl] Heinz Mfihlenbein and Dirk Schlierkamp_Voosen, *The Science of Breeding and its Application to the Breeder Genetic Algorithm*, Evolutionary Computation, 1(4), pages 335_360, 1993,

[Schoe] Eberhard Schöneburg, *Genetische Algorithmen und Evolutionsstrategien - Eine Einffihrung in Theorie und Praxis der simulierten Evolution*, Addison-Wesley Publishing Company, 1994

[Sys] G. Syswerda, *Uniform Crossover in Genetic Algoritms*, Proceedings of the Third International Conference on Genetic Algorithms, pages 2--9, In J. D. Schaffer (ed.), Morgan Kaufmann Publishers, 1989

[Whitley] D. Whitley, T. Starkweather, and D'Ann Fuquay, *Scheduling and Traveling Salesmen: The Genetic Edge Recombination Operator*, Proceedings of the Third International Conference on Genetic Algorithms, pages 133--140, In J. D. Schaffer (ed.), Morgan Kaufmann Publishers, 1989

[Rech] Ingo Rechenberg, *Evolutionsstrategie*, Fromman-Holzboog Verlag, 1973