Comparison of selection schemes for evolutionary constrained optimization

C.H.M. van Kemenade

Computer Science/Department of Software Technology

CS-R9649 1996

Report CS-R9649
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Comparison of Selection Schemes for Evolutionary Constrained Optimization

C.H.M. van Kemenade

CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
kemenade@cwi.nl

Abstract

Evolutionary algorithms simulate the process of evolution in order to evolve solutions to optimization problems. An interesting domain of application is to solve numerical constrained optimization problems. We introduce a simple constrained optimization problem with scalable dimension, adjustable complexity, and a known optimal solution. A set of evolutionary algorithms, all using different selection schemes, is applied to this problem. The performance of the evolutionary algorithms differs strongly. Selection schemes that only use a limited number of offspring as parents for the next generation consistently outperform the schemes that accept all offspring as parents and adjust their fertility based on (relative) fitness during the experiments.

AMS Subject Classification (1991): 68T20
CR Subject Classification (1991): G.1.7, I.2.8
Keywords & Phrases: evolutionary algorithms, (constrained) optimization
Note: This paper was presented at the Eighth Dutch Conference on Artificial Intelligence NAIC'96, Utrecht, The Netherlands, 1996

1. Introduction

Evolutionary algorithms are using Darwinian principles in order to evolve good solutions to certain problems. Amongst the class of EA's we have genetic algorithms [2], Evolution Strategies [5], and Evolution Programming [1]. When applying an EA we first have to find an encoding for a potential solution to our problem. In this paper we discus numerical optimization, where we have to find the vector $\vec{x}^*$ that maximizes a certain objective function $f : \mathbb{R}^d \rightarrow \mathbb{R}$. In this case a natural encoding of a solution would be a vector of real values that corresponds to a candidate solution $\vec{x}$. A candidate solution will be called an individual in the rest of this paper. Given a set of individuals, called the population, we iteratively apply the following two steps:

selection select a set of parent individuals,

production apply evolutionary operators to produce offspring individuals from these parents.

The selection step should enforce a certain selective pressure in order to guide the population towards regions of the search space containing high quality solutions. Such a pressure is obtained by introducing a bias towards better performing individuals during selection. As a result better individuals are more likely to produce offspring and more emphasis will be put on those parts of the search-space containing these individuals. So the selection step should result in the exploitation of the information present within the current population of individuals.

Exploration (creation of new candidate solutions) is provided by the production step. Given a set of parents the evolutionary operators are used to produce offspring individuals. These offspring individuals should be different from the parents. Basically we have two classes of evolutionary operators.

The first class contains the mutation operators. These operators take one parent individual and the offspring individual will be a copy of the parent individual with a (random) perturbation added. In case of a binary encoding such a perturbation can be the inversion of a single bit, and in a real-valued domain this perturbation can be implemented by the addition of some Gaussian distributed noise. So mutation operators implement a (randomized) local search in a neighborhood around the parent individual. The second class of evolutionary operators contains the recombination operators, also called crossover operators. Such operators require at least two parents. An offspring individual is created by combining parts from different parents. For example, in case of a binary encoding and two parents we can create an offspring individual by taking half of the bit-values from the first parent, and the other half from the second parent. So recombination tries to combine values from different parents in hope to find a superior solution.

In this paper we are going to focus on the application of EA's to numerical constrained optimization problem. This kind of problems will be described in section 2. Section 3 describes a specific trap that EA's can easily fall into and introduces a simple numerical constrained optimization problem containing this kind of trap. During our experiments we studied the effectiveness of selection schemes used in genetic algorithms and in evolution strategies. The different selection schemes are discussed in section 4. The results are presented in section 5, followed by the conclusions in section 6.

2. Numerical constrained optimization

Many numerical constraint optimization problems (NCOP) can be written in the form:

$$\begin{array}{rll} \text{objective:} & \text{MAXIMIZE } f(\vec{x}) & \\ \text{constraints:} & x_{i,low} \leq x_i \leq x_{i,high} & \text{(box)} \\ & G(\vec{x}) \leq 0 & \text{(inequality)} \\ & H(\vec{x}) = 0 & \text{(equality)} \end{array}$$

where $\vec{x} \in \mathbb{R}^d$ is called the objective vector, $f(\vec{x})$ is the objective function, $G(\vec{x}) \leq 0$ denotes the set of inequality constraints of type $g(\vec{x}) \leq 0$ and $H(\vec{x}) = 0$ denotes the set of equality constraints of type $h(\vec{x}) = 0$. The goal is to find the objective vector $\vec{x}^*$ that maximizes the objective function and simultaneously satisfies all the constraints. Note that all function optimization problems can be written as a NCOP having just box-constraints, so the class of function optimization problems is a subset of NCOP.

The complexity of the constraints have a strong influence on the difficulty of NCOP's. Linear constraints, such as box-constraints are relatively easy to process, as such constraints give a feasible region consisting of a single convex hull. Non-linear constraints can result in an irregularly shaped feasible region, which might even be disconnected.

When applying evolutionary algorithms to a NCOP, one has to cope with the constraints. These constraints can be handled by means of penalty functions, decoders, or repair operators [3]. Here we use penalty functions to recast the original problem to a new problem having only box-constraints and using the modified objective:

$$f^{(1)}(\vec{x}) = f(\vec{x}) - \alpha \cdot P(\vec{x}, G, H).$$

The function $P(\vec{x}, G, H)$ is assumed to be a measure of the number of constraints in the sets G and H that are violated for objective vector $\vec{x}$, and α is a multiplier that balances the relative strength of the objective $f(\vec{x})$ and the penalty $P(\vec{x}, G, H)$. Choosing an appropriate value for α will probably require some kind of "black-magic". This value will depend amongst other things upon the maximal values of $f(\vec{x})$ and its set of partial derivatives $\partial f(\vec{x})/\partial x_i$. When handling a black-box NCOP it is not known whether the regions of high overall fitness do intersect with the feasible region. If this is not

the case the objective $f^{(1)}(\vec{x})$ can guide the search away from the feasible region for low values of α, hence reducing the probability that the optimum is found.

In order to prevent the necessity of choosing a value α we can use the following objective:

$$f^{(2)}(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if G and H satisfied} \\ -P(\vec{x}, G, H) & \text{otherwise.} \end{cases}$$

When optimizing this function the original objective $f(\vec{x})$ is only calculated if none of the constraints is violated. For some NCOP's this might even be a requirement, as the original objective can be undefined when constraints are violated.

It is possible to introduce gradient information in $P(\vec{x}, G, H)$ by also incorporating an additional term indicating how strongly the constraints are violated. The search process can benefit from such gradient information when comparing two individuals violating the same number of constraints.

It is important that this gradient term can not get too large, while this might result in a competition between constraints. Therefore we propose:

$$P(\vec{x}, G, H) = \sum_{g \in G} p(g(\vec{x})) + \sum_{h \in H} p(-|h(\vec{x})|)$$

where

$$p(y) = \begin{cases} 1 + (1 - e^{-y/\gamma}) & \text{if } y \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

where γ is a scaling factor. When using this $P(\vec{x}, G, H)$ all violated constraints will result in a contribution within between one and two. Figure 1 shows a simple inequality constraint $g(x)$, and its corresponding penalty $P(x, g)$.

3. Regions of attraction

The region of attraction of a local optimum $\vec{z}$ is defined as the largest set of points $attr(\vec{z}) \subseteq \mathbb{R}^d$, such that for any starting point $\vec{y} \in attr(\vec{z})$ the infinitely small step steepest ascent algorithm will converge to this local optimum $\vec{z}$ [7]. The relative sizes of the regions of attraction of different local optima can strongly influence the behavior of EA's.

Although there is a large variety of different EA's, they all seem to have one property in common, i.e. that all these methods focus the search on the regions having a high overall fitness. Within these regions better solutions are searched for. EA's show the tendency to prefer large regions having a high average fitness [4], as such regions are discovered more easily than small regions, or narrow peaks. This preference for large regions above narrow peaks even gets magnified by applying recombination [8]. When the global optimum is present within one of the regions focussed on, this decision is correct, but if the region of attraction of the global optimum does not coincide with one of these regions, the EA is likely to converge to a suboptimal solution.

Within constrained numerical optimization problems it can easily happen that the optimum is located in a narrow peak, even if the objective function is a smooth function. In the rest of this paper we are going to study the performance of different evolutionary algorithms on the stepping stone problem (SSP-a), which is defined by:

$$\begin{array}{rl} \text{objective:} & \text{MAXIMIZE } \sum_{i=1}^{d} (x_i/\pi + 1) \\ \text{constraints:} & -\pi \leq x_i \leq \pi \\ & \exp x_i/\pi + \cos(2 \cdot x_i) - 1 \leq 0. \end{array}$$

The objective is linear, and without the constraints this would be a very simple optimization problem. The dimension of the problem d is a free parameter. By increasing this parameter the problem gets

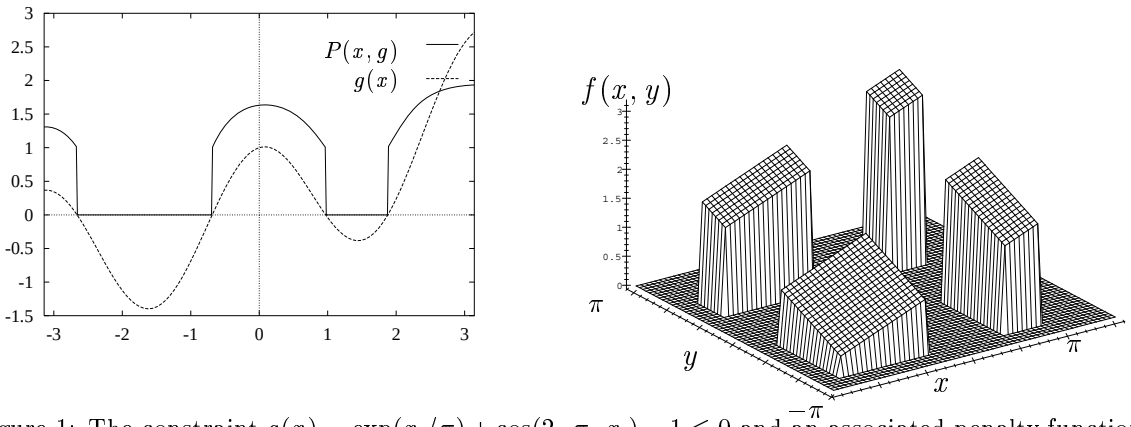

Figure 1: The constraint $g(x) = \exp(x_i/\pi) + \cos(2 \cdot \pi \cdot x_i) - 1 \leq 0$ and an associated penalty function (left) The stepping stone problem SSP-a in two dimensions

more difficult. Figure 1 shows a two-dimensional version of the SSP, where the objective function is assumed to be zero if any of the constraints is violated. The feasible region of the SSP is split in 2^d parts, which will be called stones. The relative size of a stone containing the optimal value along n dimensions, when compared to the size of the search-space, is approximately,

$$\frac{0.906^n \cdot 1.973^{d-n}}{(2 \cdot \pi)^d} \approx \left(\frac{1.973}{2 \cdot \pi}\right)^d \left(\frac{1}{2}\right)^n.$$

So for each additional x_i having the correct sign the area of the corresponding stone is reduced by a factor 2.

The optimal solution is located in a narrow peak. The region of attraction of this peak can be extended a little bit by using an appropriate penalty function, but if the dimension of the problem d rises, the region of attraction of the optimal solution inevitably becomes very small compared to the volume of search space.

The SSP is specifically designed to show a certain weak point of evolutionary solvers, i.e. its preference for broad peaks. On the other hand however it also shows a possible strong point of population based optimization techniques. The SSP has a disconnected feasible region. As a result one can not assume the existence of a completely feasible path from a current feasible solution to the global optimum. This causes problems to some path-oriented optimization techniques, that only use one sample simultaneously. An additional advantage of evolutionary techniques using recombination is the implicit parallelism that occurs when several dimensions can be optimized independently of one another, as in the SSP.

A second stepping stone problem (SSP-b) is introduced. This problem has the same set of constraints as the SSP-a, but a different objective function has to be maximized, i.e.

$$f(\vec{x}) = \begin{cases} \sum_{i=1}^{d} (x_i/\pi + 1) & \text{if all } x_i \geq 0 \\ 1 & \text{otherwise.} \end{cases}$$

Figure 2 shows a two dimensional version of this problem. The SSP-b is assumed to be more difficult than the SSP-a, as its objective function gives less information regarding the location of the optimal solution when we are still far away from this optimum. When the candidate solution $\vec{x}$ is outside the stone containing the optimum a constant value is returned.

The stepping stone problems are academic problems with a simple definition. Interesting numerical constrained optimization problems usually contain a more complex set of constraints, and/or a more

difficult objective function. But the type of constraints that occur in the stepping stone problems can also be observed in real optimization problems. So if an optimizer can not handle the stepping stone problems, this is a strong indication that it will also have problems with real NCOP's.

4. Evolution schemes

In order to get a proper comparison between different selection schemes we define a common setting for the other parts of the EA's, besides the selection scheme. The main evolutionary operator is both a recombination and a mutation operator. For each dimension it chooses the value of one of its parents with equal probability and adds some Gaussian distributed noise to it. Given two different parents $\vec{x}^{(p1)}$ and $\vec{x}^{(p2)}$, an offspring is created according to

$$x_i^{(o)} = (x_i^{(p1)} \textbf{or } x_i^{(p2)}) + N(0, \sigma)$$

where $N(a, \sigma)$ is a normally distributed random variable with $\sigma = |x_i^{(p1)} - x_i^{(p2)}|/3$.

Furthermore a discrete recombination operator is applied with a low probability $P_{discrete}$. The discrete recombination operator creates a value for the offspring using the formula

$$x_i^{(o)} = x_i^{(p1)} \textbf{or } x_i^{(p2)}.$$

All algorithms use a generational approach and a similar population size P_{size}. The following selection schemes are compared:

tournament-selection: Parents are selected using tournaments of size τ. P_{size} offspring are generated, which will be the parents during the next generation.

ES-selection: The parents are drawn uniformly from a population of P_{size} parent individuals. This proces is repeated until λ offspring individuals have been produced. Next a new parent population is created by selecting the P_{size} best individuals amongst these λ offspring.

truncE-selection: P_{size} offspring are created using uniform selection of the parents. Next the P_{size} best individuals amongst parents and children are selected as parents for the next generation.

cluster-selection: A two-stage selection is applied. First a truncE-selection done, followed by a clustering of the selected individuals. For each cluster the best performing individual is selected as a representative. Only these representatives are allowed to reproduce.

The tournament-scheme is often used in generational Genetic Algorithms [2]. The ES-scheme corresponds to (μ, λ) selection scheme with $\mu = P_{size}$, as used in Evolution Strategies [5]. The truncE scheme corresponds to the $(\lambda + \lambda)$ evolution strategy. The cluster selection scheme is a modified $(\lambda + \lambda)$ Evolution Strategy that uses a clustering step to extract additional information regarding the search space [8]. Many evolutionary algorithms, especially those using recombination, tend to create new individuals mainly in those parts of the search space that already contain many individuals. As a result clusters of individuals tend to attract the complete population. This can easily lead to premature convergence. The cluster selection scheme tries to prevent this type of premature convergence by reducing each cluster to a single well-performing representative. Although this scheme does allow efficient usage of local optimization, we did not use this option during the experiments because we are mainly interested in the behavior of the different selection schemes.

5. Experiments

During all experiments we use the following parameter settings. The scaling factor for the penalty-function $\gamma = 1$, the population size $P_{size} = 100$, the probability that discrete recombination is applied

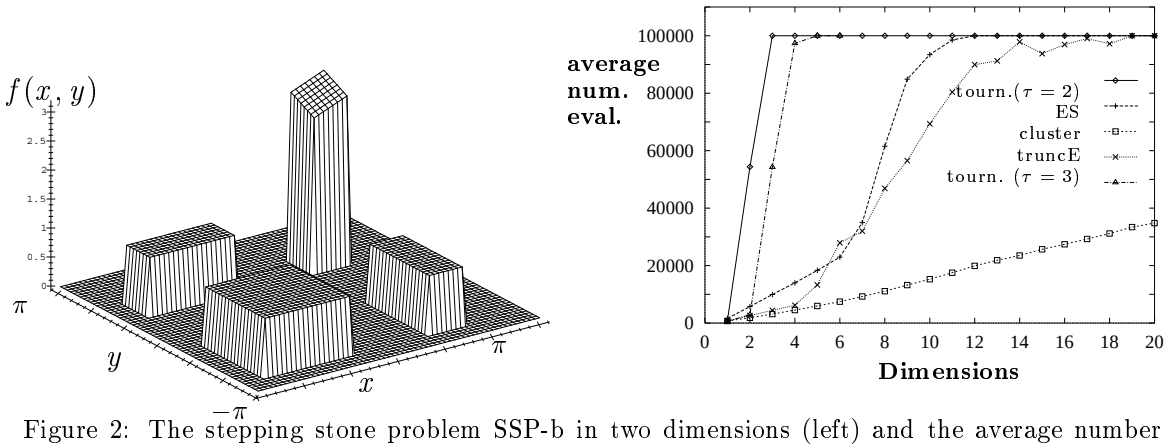

Figure 2: The stepping stone problem SSP-b in two dimensions (left) and the average number of evaluations used to located the optimum on the SSP-a (right)

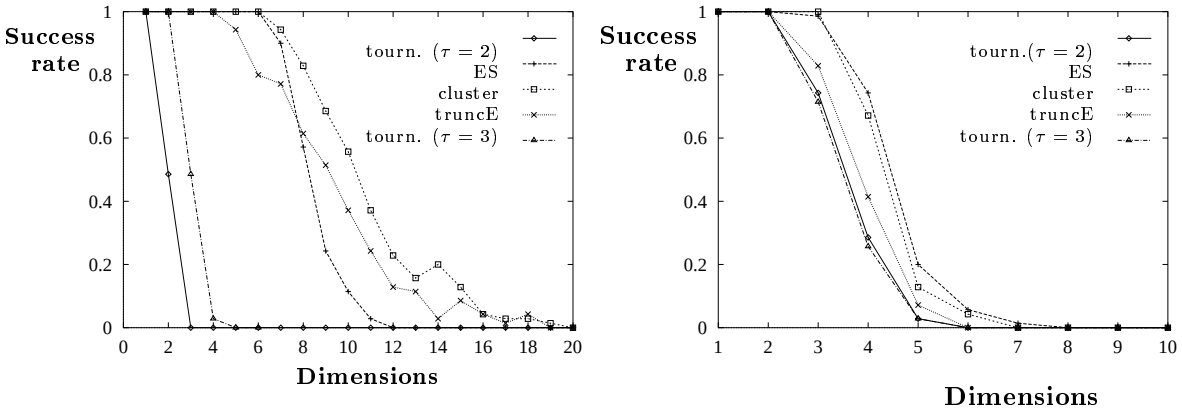

Figure 3: Percentage of runs that located the optimum on the SSP-a (left) and the SSP-b (right)

$P_{discr} = 0.1$, the number of offspring per parent in ES-scheme is set to $\lambda = 7$, the tournament size in tournament-scheme is set to $\tau = 2$ or 3, and the maximal number of function evaluations during a single run is set to 100,000. During the experiments the dimension of the problem is varied in between 1 and 20. The results are all averaged over 70 independent runs.

The right graph of Figure 2 shows the average number of function evaluations used. Tournament-scheme performs poor, and cluster-scheme is performs best. Even when it does not converge any more the cluster-scheme is usually able to terminate before it reaches the maximal number of function evaluations. This is due to the fact that it can detect an unsuccessful run as all clusters get a diminishing size. But also in case of a successful run the cluster-scheme is efficient in terms of the number of function evaluations needed.

Figure 3 shows the percentage of runs that located the optimum as a function of the dimensionality d of the problem. A run is said to be successful if $|f(\vec{x}^*) - f(\vec{x}')| \leq 10^{-3}$ for the obtained solution $\vec{x}'$, where $\vec{x}^*$ is the location of the global optimum. We see that the EA using tournament selection performs poor. The other three schemes perform consistently better.

The left graph of Figure 3 shows the results when applying the different schemes to the SSP-a. The tournament of size three performs better than a tournament size of two. Therefore the poor

performance of the tournament-scheme must probably be attributed to its relatively low selective pressure. The ES-scheme gets its selective pressure due to small fraction of well-performing offspring that are allowed to become parents. The truncE-scheme and the cluster-scheme get their selective pressure from the population-elitism incorporated.

The right graph of Figure 3 shows the results when applying the different schemes to the SSP-b. Please note that the range of the horizontal axis differs between two graphs. In this problem the objective function gives less information. As soon as one of the variable x_i gets below zero the objective function returns a constant value. The best results are still observed for the ES-scheme, clust-scheme and truncE-scheme. An interesting observation is made when comparing the success rates to the success rates on the SSP-a. All schemes, except for the tournament scheme, experience a severe decrease in success rate, when making the transition from the SSP-a to the SSP-b. The performance of the tournament scheme is better on the SSP-b. This difference is caused by the parent replacement policies of the different schemes. Within the tournament selection scheme all offspring will become parents, and selective pressure is due to the fact that some parents are more fertile than others. Within the other schemes the selective pressure is due to the fact that only the best performing individuals will become parents, but there is no difference in fertility between parents. Schemes where the selective pressure resides in the replacement policy, like ES-, clust-, and truncE-scheme are less sensitive to premature convergence [9].

When the tournament-scheme is used, the population converges rapidly. If the optimum is not located within this short period we have premature convergence. Using a larger tournament-size seems to pay of because this enlarges the rate of success. Tournament selection performs better on the SSP-b because the population converges slower on this problem. This is a result of the fact that all suboptimal peaks have exactly the same height, so none of them is preferred.

The clustering scheme uses a clustering step during selection in order to get a better sampling of the search-space and to decrease the preference of the EA for optima having large regions of attraction. This is an important advantage as it is not only important to find the different parts that constitute the optimal solution. The algorithm should also get the opportunity to mix all those parts to construct the actual optimal solution [6]. A high selective pressure can easily result in dense clusters of well-performing individuals that recombine easily. Such clusters can result in a loss of alleles that are needed to find the optimum, but that are not typical for these clusters. This corresponds a kind of cross-competition between alleles, which seems to be difficult to prevent if one does not consider distributional aspects during the selection.

6. Conclusions

The stepping stone problem (SSP-a) is not very difficult. The regions of high fitness have a relatively small region of attraction, but the objective function is simple and guides the population straight to the optimum. An EA having a high selective pressure can solve this problem, as long as the dimension of search space is not too large.

When the objective is more complex, such as in the SSP-b, the problem becomes more difficult. A selective pressure that is too strong can easily lead to premature convergence, and hence to a sub-optimal solution. Current results suggest a clear advantage for selection-schemes that induce selective pressure by using only a limited number of offspring as parents of the next generation over selection schemes that accept all offspring and adjust the fertility based on (relative) fitness of the individuals. This can be especially important in the domain of constrained optimization where one has to cope with rugged fitness landscapes, that easily result in the production of inferior offspring due to the constraints. Eliminating this inferior offspring by using large tournament sizes in a generational genetic algorithm often leads to premature convergence. Although a high selective pressure does reduce the influence of inferior offspring, it also induces large differences in fertility between well performing individuals.

The cluster-selection seems to have pleasant properties. It has a high selective pressure and the application of the clustering step and the selection of representatives can help in preventing the oversampling of certain parts of the search-space and hence premature convergence. Within the current experiments the cluster-selection performs best. It can handle more difficult problems than the other selection schedules, and it does so using less function evaluations.

We think that relatively simple constrained optimization problems, like the stepping stone problem, can help in getting a better understanding of how EA's behave when applied to numerical constrained optimization problems. The results presented in this paper are also relevant for unconstrained optimization problems, but in that case the observed effects might be less strong, as unconstrained optimization problems tend to be smoother.

Further research will be devoted to finding a more rigid mathematical foundation for our present results, and to the study of performance of EA's on problems having more complex constraints and objectives.

Acknowledgement: I would like to thank anonymous referees and Joost N. Kok for their useful comments.

REFERENCES

1. D.B. Fogel. *Evolving artificial intelligence.* Doctoral dissertation, University of California, 1994.
2. D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.
3. Z. Michalewicz and C.Z. Janikow. Handling constraints in genetic algorithms. In *Fourth International Conference on Genetic Algorithms*, pages 151–157, 1991.
4. H. Mühlenbein and H.M. Voigt. Gene pool recombination in genetic algorithms. In I.H. Osman and J.P. Kelly, editors, *Metaheuristics international Conference, Norwell.* Kluwer Academic Publishers, 1995.
5. H.-P. Schwefel. *Evolution and Optimum Seeking.* Sixth-Generation Computer Technology Series. Wiley, New York, 1995.
6. D. Thierens and D.E. Goldberg. Mixing in genetic algorithms. In S. Forrest, editor, *Fifth International Conference on Genetic Algorithms*, pages 38–45. Morgan Kaufmann, 1993.
7. A. Törn and A. Žilinskas. *Global optimization.* Lecture Notes in Computer Science 350. Springer, 1989.
8. C.H.M. van Kemenade. Cluster evolution strategies, enhancing the sampling density function using representatives. In *Third IEEE conference on Evolutionary Computation*, pages 637–642, 1996.
9. C.H.M. van Kemenade, J.N. Kok, and A.E. Eiben. Controlling the convergence of genetic algorithms by tuning the disruptiveness of recombination operators. In *Second IEEE conference on Evolutionary Computation*, pages 345–351. IEEE Service Center, 1995.