A two-level evolution strategy (balancing global and local search)

C.H.M. van Kemenade

Computer Science/Department of Software Technology

CS-R9559 1995

Report CS-R9559
ISSN 0169-118X

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

A Two-level Evolution Strategy

balancing global and local search

C.H.M. van Kemenade

CWI

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

kemenade@cwi.nl

Abstract

Evolution Strategies apply mutation and recombination operators in order to create their offspring. Both operators have a different role in the evolution process: recombination should combine information of different individuals, while mutation performs a kind of random walk to introduce new values. In an ES these operators are always applied together, but their different roles suggest that it might be better to apply them independently and at different rates. In order to do so the ES has been split into two levels. The resulting Modular Evolution Strategy consists of a population of local optimizers and a distributed population manager. Both parts have their own specific role in the optimization process. As a result of its modularity this method can be adapted more easily to specific classes of numerical optimization problems, and introduction of adaptive mechanisms is relatively easy. A further interesting aspect about this algorithm is that it does not need any global communication, and therefore can be parallelized easily.

Many problems can be expressed as numerical optimization problems. Especially when the dimension of the input space and the number of local optima is high these problems tend to be very difficult. In order to obtain an efficient solver one has to gather information regarding the function to be optimized. Evolution based learning can be used to obtain this information. This paper contains results obtained with the Modular Evolution Strategy and compares these results to those obtained with other evolution based method. The results look promising.

AMS Subject Classification (1991): 68T20
CR Subject Classification (1991): G.1.7, I.2.8
Keywords & Phrases: genetic algorithms, optimization
Note: Paper is to be presented at the 5th Belgian-Dutch Conference on Machine Learning, Brussels 1995

1. Introduction

Evolution based learning systems, such as Genetic Algorithms, Genetic Programming, Evolution Programming, and Evolution Strategies have many important areas of application. A problem can be difficult for several reasons. Some problems are not well defined, as their constraints vary in time, as they are overconstrained, or as they involve fuzzy constraints. Other problems are difficult as almost nothing can be assumed regarding structures present in their search space. On such problems one needs a problem solver that can learn about the structures present the search space. Furthermore a certain amount of robustness and adaptiveness is needed when the constraints vary in time or the structures of the search space change as one moves to another part or the resolution one looks at is modified. Evolution has shown to be successful on extremely difficult optimization problems in nature. Many successful applications of simulated evolution exist. It might be tempting to mimic biological evolution as closely as possible. But evolution in nature has other goals, and other restrictions than simulated evolution. New and interesting alternatives to the current Evolution based learning systems might be obtained by deviating even further from the path taken by natural evolution. Currently we are working on a modified form of Evolution Strategies (ES) [Rec73, BSM93, Sch95]. Recombination and mutation have different aims in ES. Recombination is used to combine information available in different individuals, while mutation is essentially a randomized way of hill-climbing. In the $(\mu \overset{+}{,} \lambda)$-ES recombination and mutation are applied simultaneously. In order to get a better separation between the different tasks of recombination and mutation we have defined a two-level ES, the Modular Evolution Strategy (MES). In MES we have a population of mutation based local hill-climbers. At the second level we have a population manager that uses a recombination operator to combine information and add global search.

Our first tests show results comparable to those obtained by the $(\mu \overset{+}{,} \lambda)$-ES. As the MES still contains a few parameters which can be optimized, we expect to be able to outperform the $(\mu \overset{+}{,} \lambda)$-ES.

Another interesting property of the MES is that it can do all its computations in a distributed manner. The standard $(\mu \overset{+}{,} \lambda)$-ES still needs global communication after each generation in order to determine the new pool of parents.

Section 2 describes the numerical optimization problem and discuses when the evolution based learning methods might be the major player to solve these kinds of problems. In the section 3 a brief introduction to Evolution Strategies is given. Section 4 describes the Modular Evolution Strategy. The experimental setup is described in section 5. In section 6 our results are shown and compared to results obtained with other evolution based methods [BSM93, Bäc94, WGM94, PJ94, EvKK95]. Section 7 contains conclusions and directions for further research.

2. Numerical optimization problem

The numerical optimization problem involves the search for the global optimum of a function. To be more specific:

$$\begin{array}{l} \textit{Given a function } f : I\!R^d \rightarrow I\!R \\ \textit{find the vector } \vec{x} \in I\!R^d \textit{ such that} \\ \forall\, \vec{y} \in I\!R^d \ : \ f(\vec{y}) \leq f(\vec{x}) \end{array}$$

When f is a relatively smooth, low dimensional function there are many methods to solve this problem. For one-dimensional functions f it is often possible to determine this maximum analytically. Even when such an analytical solution can not be found, it is often possible to find solutions by means of covering methods, that sample a bounded interval. When the dimension d of the input space increases the problem gets much more difficult. If the derivative $\partial f(\vec{x})/\partial x_i$ exists for all dimensions gradient-descent methods might be used. A gradient-descent method is an iterative method using the derivatives of f to find a direction of (maximal) profit and moves in this direction to increase f. Such gradient-descent methods get stuck in local optima easily. If the number of local optima is not too large the global optimum can be found by running a number of independent gradient-descent solvers. As d increases, the volume of the search space and the number of local optima often increase exponential in d. Gradient-descent methods break down on such large numbers of local optima and more advanced methods have to be applied. Several search methods exist to locate the global optimum in such cases, each having there own advantages and disadvantages [TŽ89]. All these methods have in common that these try to learn properties of the fitness landscape, the function to be optimized, during the search process. This information is needed to guide the search in the proper direction and bound the computational power needed to locate the global optimum.

We are specifically interested in probabilistic search methods as randomization can be used to obtain efficient and robust methods to handle a diverse suit of numerical optimization problems. In the class of probabilistic optimization methods we have, amongst others,

- Monte-Carlo methods,
- Simulated Annealing [KGV83, AK89],
- Genetic Algorithms [Hol75, Gol89, BSM93],
- Evolution Programming [Fog94, BSM93], and
- Evolution Strategies [Rec73, BSM93, Sch95].

Whether Genetic Algorithms should be part of this list is a topic of discussion. According to DeJong GA's are no function optimizers [Jon93]. On the other hand there are many successful applications of GA's to numerical optimization problems [Whi89, WGM94, PJ94, EvKK95]. In all of these applications additional features are added to the GA, such that these GA's do not correspond to the pure definition of the GA any more.

In this paper we are going to focus on evolution based optimization methods. Evolution can be used to learn about a function landscape in a simple, elegant, and distributed way. Furthermore we are interested in possibilities to create a distributed function optimizer, as massive parallel computer are becoming more important and Local Area Networks containing numerous PC's or workstations are widely available nowadays.

3. Evolution Strategies

The first and most simple Evolution Strategy (=ES) is the two-membered (1+1)-ES developed by Rechenberg in 1964 [Rec73]. When applying this strategy a single parent is used. This parent consists of a vector $\vec{X}^{(p)}$of d real numbers that encode a possible solution to the numerical optimization problem. A single offspring is created using the formula,

$$X_i^{(o)} = X_i^{(p)} + N(0, \sigma) = N(X_i^{(p)}, \sigma)$$

where $N(a, \sigma)$ is the Gaussian or Normal distribution. The offspring replaces the parent if it outperforms the parent on the function f to be optimized. The value of σ is adjusted by means of the 1/5-success rule, which states:

> **1/5-success rule:** *determine the ratio of the number of successful mutations to the total number of trials. If the ratio is greater than 1/5 the variance σ should be increased, if it is less than 1/5 than decrease the variance σ.*

Schwefel developed the multi-membered $(\mu \overset{+}{,} \lambda)$-ES. Instead of a single parent a pool of μ parents is used. This pool of parents is used to create a set of λ offspring by means of recombination and mutation. Next a new pool of parents has to be selected, that will be used to create the next generation. In a $(\mu + \lambda)$-ES the best μ individuals from both parents and offspring are selected, and in the (μ, λ)-ES the new parents are selected from the set of λ offspring only. Usually the ratio $\mu/\lambda = 1/7$. Several recombination operators exist for the $(\mu \overset{+}{,} \lambda)$-ES. The two most important are the discrete and the intermediate recombination. The discrete recombination is defined as

$$X_i^{(o)} = X_i^{(p_1)} \text{or} X_i^{(p_2)},$$

which means that the value of each field is taken either from the first or from the second parent. The second important recombination operator is the intermediate recombination defined as

$$X_i^{(o)} = \chi X_i^{(p_1)} + (1 - \chi) X_i^{(p_2)},$$

where χ is a random value in range $[0, 1]$, which corresponds to a randomized linear interpolation between the two parents.

Furthermore each individual in the $(\mu \overset{+}{,} \lambda)$-ES contains apart from its objective vector $\vec{X}$ a vector $\vec{\sigma}$, containing one value σ_i for each dimension. The 1/5-success rule, used for the (1+1)-ES, can not be used for such a vector. So the $(\mu \overset{+}{,} \lambda)$-ES adjusts $\vec{\sigma}$ by means of a log-normal distributed mutation operator in the evolution process. A more detailed treatment of Evolution Strategies can be found in [BSM93, Sch95].

4. Modular Evolution Strategy

The modular Evolution Strategy (MES) is based on the existing Evolution Strategies, as described in section 3, and previous experiences in application of Genetic Algorithms to numerical optimization problems [EvKK95, vKE95]. The goal of our research is to develop a simple, efficient and modular solver for optimization problems. We do this by the design of a two-level ES, resulting in a more clear separation between the tasks of recombination and mutation in the search process. In the rest of this section the design of the MES will be motivated in a number of steps. The design is focussed on finding simple, elegant extensions of the algorithm in order to keep the resulting algorithm as simple and clean as possible. At the end of this section the complete algorithm of MES will be presented.

The prime building block of the MES is a local optimizer LocOpt that, given an initial starting point, rapidly moves towards a nearby optimum. This LocOpt is not expected to locate the global optimum. Furthermore the LocOpt should be robust in the sense that it makes as few assumptions as possible regarding the actual shape of fitness landscape it is traveling through. Given these requirements we chose to use a (1+1)-ES as LocOpt, but other choices are possible as well. When additional information regarding the shape of the fitness landscape is available, this information might be used to select another LocOpt. For example if one knows the fitness landscape is smooth and does not contain noise, it might be interesting to use a gradient descent method as LocOpt.

Usually the LocOpt does not converge to the global optimum. One way to obtain the global optimum might be to run the LocOpt many times, each time starting from another initial location. This approach might work for a fitness landscape containing only a small number of local optima, but when the number of local optima increases the efficiency of this method decreases rapidly.

A more efficient search might be obtained by running a population of LocOpt's simultaneously. After some time we can reduce the set of LocOpt's by discarding those LocOpt that having the lowest fitness, as these can be assumed to move towards an inferior local optimum. As the algorithm proceeds the number of LocOpt is decreasing and more computation is spend on a smaller part of the total search space, resulting in a significant benefit over the previous method, where all LocOpt's converge to a local optimum. When the number of local optima is large, this method still needs a large number of initial LocOpt's, and therefore requires a large computational effort.

It would be nice if we could start with a relatively small number of LocOpt and later add more diversity to the search, by enlarging this number of LocOpt, whenever needed. In order to do so efficiently, we need to exploit the structures present in the fitness landscape. Often there is a (weak) relation between optima in the sense that the good local optima tend to appear in clusters instead of being randomly spread over the search space. One way to exploit this kind of relation is by means of a recombination operator that creates a new LocOpt based upon the location of two parent-LocOpt's using the formula

$$X_i^{(o)} = \chi_i \, X_i^{(p_1)} + (1 - \chi_i) \, X_i^{(p_2)},$$

where χ_i is a random value in range $[0, 1]$. This operator corresponds to selecting a new initial location within the d-dimensional hypercube spanned by the two parents.

A straight forward way to add recombination would be:

1. let all LocOpt do a number of iterations
2. divide the population in disjunct subsets all containing exactly three LocOpt's
3. in each subset, apply the recombination operator using the two LocOpt's having the highest fitness as parents
4. in each subset, let the offspring LocOpt replace the worst LocOpt

So each time recombination is applied 1/3 of the original population will be replaced by offspring.

Under many circumstances this will work out fine. Initially, when the parents are spread over the complete search space, the offspring can also arise anywhere, but when the population of LocOpt's are concentrated in a sub region of the complete search space, the created offspring will be too. So the combination of selection and reproduction will result in the search being restricted to a sub region of the complete search space, thereby resulting in a more efficient search. On many problem instances this method seems to perform well, but on certain problems the method seems to fail to find the global optimum. The simplest solution to this problem is increasing the size of the population. The effect of this increase is twofold, ie.

- the introduction of more LocOpt's results in a denser sampling of the search space, and therefore a better exploration of this space, and
- the take-over time, which is a measure of the time it takes a well performing LocOpt to spread a large number of almost identical copies and thereby taking over the complete population of LocOpt's, will increase

Disadvantage of increasing the size of the population is that we return to the old situation, where a lot of computation is needed due to the high number of LocOpt that have to run.

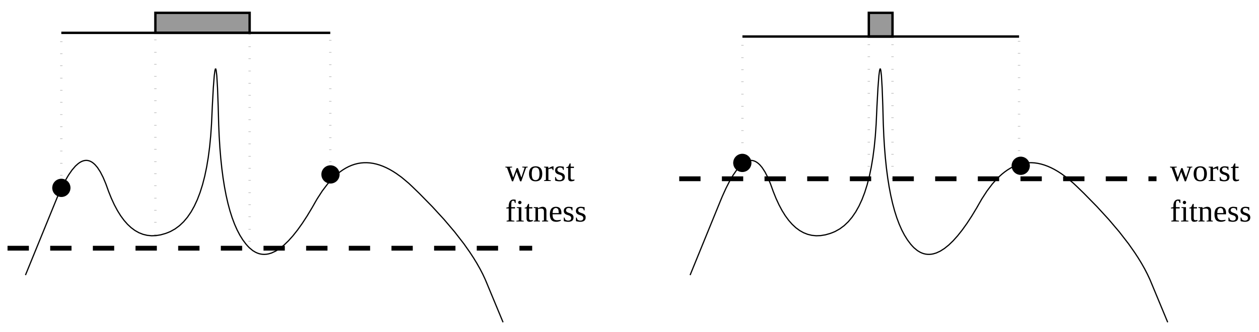


Figure 1: Effect of recombination

When taking a closer look at the type of fitness landscapes that cause trouble to this method, it often appeared to be fitness landscapes containing many relatively sharp peaks of approximately same height. When searching the global optimum is this type of landscapes we see that the use of recombination pays off in the beginning, but when the average fitness gets closer to the optimum the convergence stagnates. These effects can be understood when taking a look at the probability that recombination results in a new LocOpt that outperforms the current worst LocOpt in the population. Figure 1 shows a simple example of a 1-dimensional fitness landscape. The left graph shows a snapshot taken when the search still proceeds well. The two black dots indicate location of the two parent

LocOpt's and the dotted line indicates the fitness of the worst LocOpt present within the population. The solid line above the picture shows all possible locations for the offspring LocOpt. A new LocOpt is assumed to be successful if,

- it is in the region of attraction of another optimum than any of its parents, and
- its fitness is above the current fitness of the worst LocOpt within the population.

A LocOpt that violates the first restriction will probably result in the same local optimum being rediscovered. The second requirement is necessary as a LocOpt that violates this requirement will be removed from the population fast. The bar above the graph indicates the region in which the new LocOpt should be in order to be successful according to the above definition. The length of this bar with respect to the underlying line is an indication of the probability a new LocOpt is successful. In an early phase of the search process the length of the bar is mainly determined by the size of the region of attraction. The right graph in figure 1 shows the same fitness landscape at a later stage. The two parents have converged closer to their local optima, and the current worst fitness also increased. The probability a recombination results in a successful new LocOpt has decreased severally. This is mainly a result of the second requirement being violated more often.

Given the above observations it seems that relaxing the second requirement migh help in improving the search process. In order to do so it is necessary to give a new LocOpt more time to improve its fitness. The most direct way to obtain this goal is increasing the number of iterations a LocOpt is allowed to performa before it has to compete against the other members of the population. In order to do so we use a variable length of the interval between subsequent recombinations. Each time recombination takes place the length of this interval is multiplied by a predefined constant.

4.1 The MES-algorithm

In the experiments presented in this paper a $(1+1)$-ES is used as LocOpt, as this optimizer does not make many assumptions regarding the actual shape of the fitness landscape. The algorithm for a single iteration of LocOpt is:

```
$X_i^{(o)} = N(X_i^{(p)}, \sigma)$
if ($f(\vec{X}^{(o)}) \geq f(\vec{X}^{(p)})$)
  then begin
     $\vec{X}^{(p)} = \vec{X}^{(o)}$
     $\sigma = \sigma \cdot \alpha$
  end
  else
     $\sigma = \sigma \cdot \left(\frac{1}{\alpha}\right)^7$
```

where $N(a, \sigma)$ is the Normal distribution with mean a and variance σ and α is a constant that determines the speed with which σ will be adjusted.

The algorithm for the MES is:

```
forall $LocOpt$ do randomize($\vec{X}^{(p)}$)

$epoch$ = 10
while NOT ready
```

```
do begin
  forall $LocOpt$ do $epoch$ iterations
  recombine
  $epoch$ = $epoch \cdot \beta$
end
```

where `randomize()` creates a valid, random starting point. `Recombine` divides the complete population in random disjunct subsets of size three. Given such a set of three LocOpt the two LocOpt having the highest fitness are used as parents. An offspring is created using the formula

$$X_i^{(o)} = \chi_i \, X_i^{(p_1)} + (1 - \chi_i) \, X_i^{(p_2)},$$

This offspring replaces the $\vec{X}^{(p)}$ of the LocOpt having the worst fitness, within the subset. Next a new value of the σ parameter of this LocOpt is determined according to the formula

$$\sigma = \max\{ \, |\vec{X}^{(p_1)} - \vec{X}^{(p_2)}| \, , \sigma^{(p_1)}, \sigma^{(p_2)} \}$$

5. Experimental setup

For our test we used a set of standard suit of numerical optimization problems. These problems are often used to test optimization algorithms as each problem has its own specific characteristics.

The Hypersphere is defined as,

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2,$$

is relatively easy to optimize. It is often used to match theory and experiments.

The correlated hypersphere is defined as

$$f(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 .$$

Due to the correlation introduced by the second sum it is not possible to optimize the values along different dimensions independent of one another any more.

The Rastrigin function is:

$$f(\vec{x}) = \alpha n + \sum_{i=1}^{n} x_i^2 - \alpha \cos(2\pi x_i)$$

where $-5.12 \leq x_i \leq 5.12$. The global minimum of zero is at the location $\vec{x} = (0, 0, \cdots)$. The primary characteristic of this function is the existence of many suboptimal peaks.

The Ackley function is defined as:

$$f(\vec{x}) = 20 + e - 20 \exp \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right)$$

where $-30 \leq x_i \leq 30$. The global minimum of zero is at the location $\vec{x} = (0, 0, 0, \cdots)$. At a low resolution the landscape of this function is unimodal; however, the second exponential term covers the landscape with many small peaks and holes.

The Griewangk function is defined as:

$$f(\vec{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos \left(\frac{x_i}{\sqrt{i}} \right)$$

where $-600 \leq x_i \leq 600$. The global minium of zero is at the point $\vec{x} = (0, 0, 0, \cdots)$. This function has a product term introducing an interdependency between the variables. This interdependency often causes trouble to optimization method that try to optimize one function one variable at a time.

The definition of the function by Fletcher and Powell is taken from [Bäc94].

$$f(\vec{x}) = \sum_{i=1}^{n} (A_i - B_i)^2$$

$$A_i = \sum_{j=1}^{n} a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j$$

$$B_i = \sum_{j=1}^{n} a_{ij} \sin x_j + b_{ij} \cos x_j$$

where $-\pi \leq x_i \leq \pi$. For the matrices $\mathbf{A}$, $\mathbf{B}$ and $\vec{\alpha}$ we used the same random values as Bäck used. According to Fletcher and Powell there are up to 2^n local optima and the global optimum of 0 is obtained for $\vec{x} = \vec{\alpha}$. This function is assumed to be a difficult, realistic optimization problem without an artificial structure.

These functions are actually minimization problems instead of maximization problems, but that does not matter as each minimization problem can be rewritten as a maximization problem by doing an inversion.

function/problem	Abbrev.	dimension	input ranges	optimal value	hit thres.
Hypersphere	Hyp	30	[-30,30]	0	10^{-9}
Correlated Hypersphere	Chyp	30	[-30,30]	0	10^{-3}
Discrete Hypersphere	Dhyp	30	[-30,30]	0	10^{-9}
Griewangk	Grie	30	[-600,600]	0	10^{-9}
Ackley	Ackl	30	[-30,30]	0	10^{-7}
Rastrigin	Rast	20	[-5.12,5.12]	0	1
Fletcher & Powell	Flet	30	[$-\pi$,π]	0	10^{3}

Table 1: Brief description of used numerical optimization problems

Table 1 contains a brief description of the different numerical optimization problems in our test suit. The column hit thres. defines a fitness threshold. A run is assumed to be successful if the distance, in fitness, between the best found solution and the optimum is less than this threshold.

To obtain an empirical justification for our conjectures regarding the the interval between successive recombinations, three different configurations for the MES have been tested, ie.

Configuration A: LocOpt does just 1 iteration between subsequent recombination steps ($\beta = 1$)

Configuration B: LocOpt does 10 iterations between subsequent recombination steps ($\beta = 1$)

Configuration C: LocOpt does 10 iterations before the first recombination step is done. After each recombination step the length of the interval is enlarged by a factor β ($\beta = 1.15$)

In all our tests the population size is set to 45, α is set to 0.9, and the initial value of σ is set to input range/10.

In order to compare these methods a 100 independent runs are performed on each test problem.

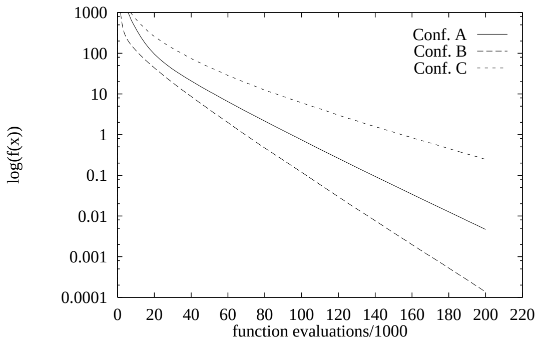
6. Results

In this section the results obtained by MES are shown. An explanation is given for the observed effects, and a comparison is made to results obtained by other evolution based methods.

Test	Conf. A			conf. B			Conf. C		
Problem	Best	Std. dev.	#hits	Best	Std. dev.	#hits	Best	Std. dev.	#hits
Hyp	5.7e-63	1.3e-62	100	1.3e-16	5.8e-17	100	4.1e-46	5.2e-46	100
Chyp	4.6e-3	2.9e-3	1	6e-4	9e-5	100	0.25	0.12	0
Dhyp	2.64	1.77	11	0.21	0.43	80	3.56	2.48	6
Grie	6.9e-3	7.9e-3	47	0.0014	0.004	88	3.9e-11	3.1e-11	100
Ackl	1.7	0.6	6	1.4e-14	1.8e-15	100	1.0e-7	5.2e-8	56
Rast	10.6	2.9	0	8.19	2.5	0	3.94	1.37	10
Flet	15e3	17e3	9	10184	9762	6	1688	1864	49

Table 2: Summary of the results obtained by MES

Table 2 shows the performance of the two methods on a number of different test problems. All results are averaged over 100 independent runs. The column Best shows the best found solution averaged over a 100 runs, the column Std. dev. shows the standard deviation in this average, and the column #hits shows the number of times a solution is found which fitness deviates less than threshold from the optimal fitness. The values of problem depedent threshold used can be found in table 1.



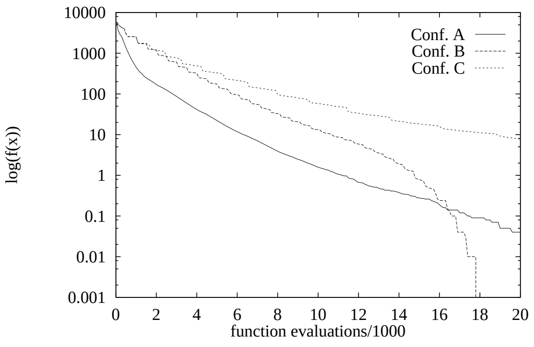


Figure 2: Average best for Chyp (left) and Dhyp (right)

Figures 2, 3, and 4 show the best fitness as a function of the number of function evaluations for the three different configurations. The curves contain averages over 100 independent runs. It is not clear which configuration performs best. In case of the hypersphere and the correlated hypersphere configuration A performs best. For the discrete hypersphere and the Ackley function configuration B performs best, and for the Griewangk, Rastrigin and Fletcher & Powell function configuration C outperforms the other two.

In case of the hypersphere and the correlated hypersphere the recombination operator is an effective operator, that can easily result in large steps being taken into promising directions. Due to the contracting effect of the recombination operator, offspring is always within the hypercube spanned by the parent, the recombination operator can converge very fast on the easy hypersphere function.

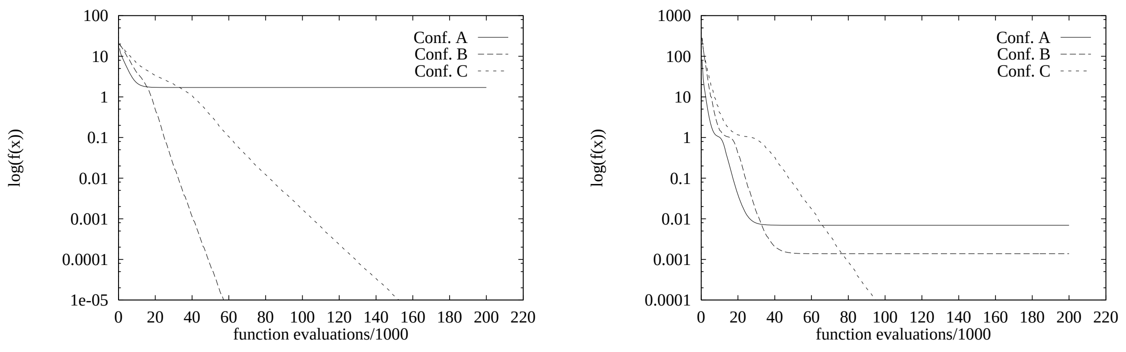


Figure 3: Average best for Ackl (left) and Grie (right)

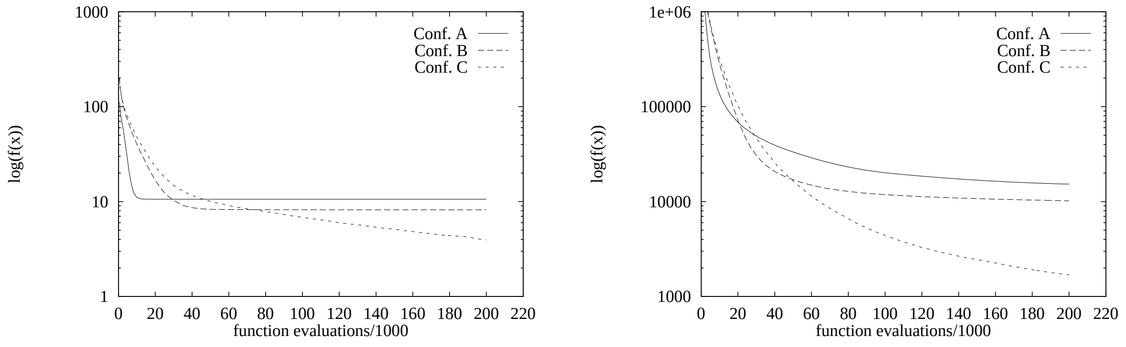


Figure 4: Average best for Rast (left) and Flet (right)

A recombination rate results in the creation of clusters of LocOpt's. Recombination between two LocOpt which are close to one another does not help in escaping local optima. Configuration B seems to offer a reasonable balance between local and global optimization. The Ackley and Rastrigin are susceptible to the same clustering problem, which in these cases also leads to a kind of premature convergence when the recombination is applied too often.

The Griewangk and Fletcher & Powell function have many peaks, resulting in offspring needing time to do some hill-climbing in order to be able determine whether these offspring have the potential to outperform their parents. For this reason configuration C performs best on these two functions.

Bäck et al. compared the performance of GA, EP, and ES on the hypersphere, discrete hypersphere, and Ackley function in [BSM93], and on the Ackley, and Fletcher & Powell function. In all those tests the ES outperformed the other two methods by far. In all instances at least one of our three configurations performs at least as well as the ES, but there is not a single configuration that outperforms the ES in on all problems. If we allow problem specific tuning of parameters we do outperform the ES on all functions. These results suggest that the MES might have the potential to outperform the ES when an appropriate adaptive mechanism is found for the MES-parameters.

Whitley et al. enhanced a GA by applying local optimizers to each offspring [WGM94]. He compares a standard GA to a GA using Lamarckian learning, where the result of the local optimizer is written back at the chromosome, and a GA applying Baldwin learning, where the optimizer is just used to correct the fitness value of an individual (make this fitness equal to the fitness of the most nearby optimum). Whitley concludes that both learning methods result in an enhanced GA. The GA using Lamarckian is faster than the GA using Baldwin learning, but the GA using Baldwin learning is more reliable. Whitley does tests on the Rastrigin (20D) and the Griewangk (20D) function. Note that our results are obtained for their 30-dimensional counterparts, which have a much larger search space. On the Rastrigin function a normal GA never finds an optimum, but when enriched with either Lamarckian of Baldwin learning it always finds an optimum. On the Griewangk function Whitley finds the optimum in about 50% of all runs, except for a test with Lamarckian learning and a population size of 400, in which case the optimum is always found. As the GA is run for 1000 generations the total number of function evaluations is equal to $4 \cdot 10^5$. On the Griewangk function MES finds the optimum in 88% of the runs within $2 \cdot 10^5$ function evaluations, which is better than the result obtained by Whitley. Whitley counts a complete run of the local optimizer is as a single function evaluation, so we expect to outperform Whitley's results by far with respect to computational effort.

Potter et al. applied a Cooperative Coevolutionary GA (CCGA) to the Rastrigin (20D), Griewangk (10D), and Ackley (30D) functions. The CCGA runs a separate GA for each dimension and assumes the values of other dimensions to be equal to the best found solution in their corresponding GA. On Rastrigin and Ackley this method can be assumed to perform well, as the different dimensions of these functions are in fact uncorrelated. On the Griewangk function this method breaks down. Probably this is the reason that Potter uses a Griewangk function having just 10 dimensions. When using a Griewangk (10D) the MES outperforms CCGA by far. On the Ackley function MES also shows approximately similar performance as CCGA does. On the Rastrigin function the CCGA performs better.

Eiben et al. used two multi-parent recombination operators to enhance GA performance [EvKK95]. They used the same set of test functions as Potter did. Again the MES outperforms the results of Eiben on the Griewangk and Ackley function, but a GA using multi-parent recombination operators outperforms MES on the Rastrigin function. When doing the comparison with respect to computational effort, the difference gets even bigger as a GA function evaluation is significant slower due to the time needed for all bitwise manipulations and genotype to phenotype mapping.

7. Conclusions

Recent results suggest that that evolution based learning is an interesting technique for solving large real-world problems. Many successful applications are obtained by combining local and global optimization in a smart way. The MES described in this paper tries to do the same for an ES. By splitting the ES in two parts, one part containing a fast, evolution-based local optimizer, and another part containing a recombination based global optimizer. Our results show that this split does make sense. Different problems require a different balance between local and global optimization. In the MES this is balance is adjusted by means of the number of iterations of the local optimizer between subsequent recombinations. We have some intuition regarding the relation between the function to be optimized and the optimal value of the interval between recombinations. Further research is needed to formalize those intuitions and design an adaptive mechanism that can learn and adjust this parameter during the evolution process. Another parameter which deserves more attention is the population size.

MES does most of its computation in a distributed way. In the current implementation the only non-distributed step is the creation of a random partition of the complete population in disjunct subsets of size 3. It is not expected to be difficult to make this step distributed too. When all steps are distributed it will be easy to exploit the full computational power of parallel computers or computer networks.

Acknowledgements: We like to thank J.N. Kok for his comments. Furthermore we thank Th. Bäck for providing the code and data of the function by Fletcher & Powell.

References

[AK89] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann machines.* Wiley, 1989.

[Bäc94] Th. Bäck. *Evolutionary Algorithms in Theory and Practice.* Doctoral dissertation, Universität Dortmund, 1994.

[BSM93] Th. Bäck, H.-P. Schwefel, and R. Männer. An overview of evolutionary algorithms for parameter optimization. *Journal of Evolutionary Computation*, 1:1–23, 1993.

[EvKK95] A.E. Eiben, C.H.M. van Kemenade, and J.N. Kok. Orgy in the computer: Multi-parent reproduction in genetic algorithms. In *Third European Conference on Artificial Life*, 1995.

[Fog94] D.B. Fogel. *Evolving artificial intelligence.* Doctoral dissertation, University of California, 1994.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[Hol75] J.H. Holland. *Adaption in natural and artificial systems.* The university of Michigan Press, 1975.

[Jon93] K.A. De Jong. Genetic algorithms are NOT function optimizers. In L.D. Whitley, editor, *Foundations of Genetic Algorithms - 2*, pages 5–18. Morgan Kaufmann, 1993.

[KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[PJ94] M.A. Potter and K.A. De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature - 3*, pages 249–257, 1994.

[Rec73] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, 1973.

[Sch95] H.-P. Schwefel. *Evolution and Optimum Seeking.* Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[TŽ89] A. Törn and A. Žilinskas. *Global optimization.* Lecture Notes in Computer Science 350. Springer, 1989.

[vKE95] C.H.M. van Kemenade and A.E. Eiben. Multi-parent recombination to overcome premature convergence in genetic algorithms. In *Seventh Dutch Conference on Artificial Intelligence*, 1995.

[WGM94] D. Whitley, V. Scott Gordon, and K. Mathias. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature - 3*, pages 6–15. Springer-Verlag, 1994.

[Whi89] D. Whitley. The GENITOR algorithm and selective pressure. In *Third International Conference on Genetic Algorithms*, pages 116–121, 1989.