Performance of Multi-Parent Crossover Operators on Numerical Function Optimization Problems

A.E. Eiben
gusz@wi.leidenuniv.nl
Leiden University
Dept. of Computer Science
P.O. Box 9512
2300 RA Leiden
The Netherlands

C.H.M. van Kemenade
kemenade@cwi.nl
CWI
Dept. of Software Technology
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Abstract

The multi-parent *scanning crossover*, generalizing the traditional uniform crossover, and *diagonal crossover*, generalizing 1-point (n-point) crossovers, were introduced in [5]. In subsequent publications, see [6, 18, 19], several aspects of multi-parent recombination are discussed. Due to space limitations, however, a full overview of experimental results showing the performance of multi-parent GAs on numerical optimization problems has never been published. This technical report is meant to fill this gap and make results available.

Contents

List of Tables

List of Figures

1 Introduction

As it was stated in earlier publications, [5, 6, 18, 19], the use of multi-parent crossovers can improve GA performance significantly on numerical optimization problems. The main goal of this report is to present the performance curves that provide empirical evidence on the advantages of multi-parent reproduction mechanisms. In particular, in this report we:

1. Give the definitions of the multi-parent recombination operators scanning crossover and diagonal crossover.
2. Present a test suite consisting of 8 difficult numerical optimization problems.
3. Discuss three performance measures:
   - success rate (the percentage of cases when an optimum is found),
   - effectivity (the best function value found by the GA),
   - efficiency (the number of evaluations before termination).
4. Present the results concerning both multi-parent mechanisms on each function according to each performance measure. As for the diagonal crossover we also make a comparison with the (2-parent) n-point crossover to show that the performance gain is not simply the result of using more crossover points.
5. Draw some conclusions and sketch our ongoing and further research.

2 Diagonal crossover

Traditional crossover creates two children from two parents by splicing the parents along the single crossover point and exchanging the 'tails'. The basic idea behind diagonal crossover is to generalize this mechanism to an n-ary $(n-1)$-point crossover. Diagonal crossover selects $(n-1)$ crossover points resulting in n chromosome segments in each of the n parents and composes n children by taking the pieces from the parents 'along the diagonals'. Figure 1 illustrates this idea.

Notice that for $n = 2$ diagonal crossover coincides with the traditional 1-point crossover. Obviously, $n-1$ crossover points specify n chromosome segments per parent, which allows the creation of n^n different children from n parents (n of which will equal the parents). By constructing children along the diagonal lines only we keep the number of offspring at 'reasonable level', in the meanwhile each child contains a segment of each parent, thus we obtain a high level of mixing genes. There are two reasons to expect that the use of more parents in diagonal crossover leads to improved GA performance: a high level of disruption and a large sample of the search space used when creating offspring. As for the first aspect, by using more crossover points the operator becomes more disruptive, thus more explorative and less sensitive for premature convergence. Secondly, by the use of more parents there is more information on the search space and there is more consensus

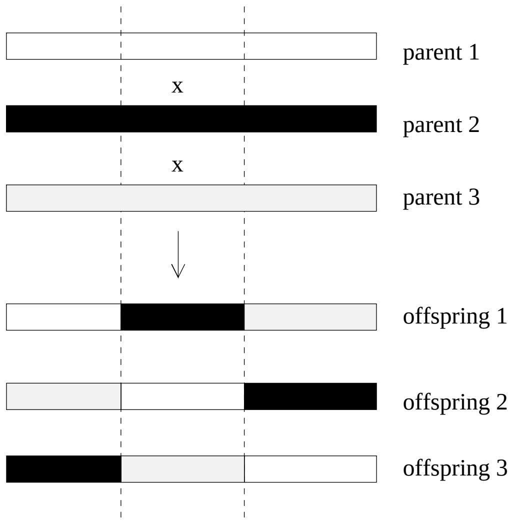

Figure 1: Diagonal crossover with three parents

needed to focus the search to a certain region, that is the danger of (too) early commitment is reduced.

Clearly, traditional 2-parent n-point crossover also has a high level of disruptivity. To see whether the usage of more parents is really important we decided to perform parallel tests with n-point crossover and monitor GA performance for different n's.

3 Scanning crossover

The idea behind scanning crossover is to take n parents and to create one child by 'scanning' the parents' genes deciding at each gene which parent can deliver its allele to the child. The pseudo-code for scanning crossover is the following.

INITIALIZE MARKERS
(% mark the 1st gene in each parent, open/mark position 1 in the child)
FOR $child.marker = 1$ TO $chrom.length$
    CHOOSE one marked gene from the parents and include it in the child
    UPDATE markers
END FOR

Obviously, traditional uniform crossover works by the same mechanism, always making a random choice. In scanning crossover the choice mechanism is not defined in general. It can be a uniform random choice (uniform scanning), it can choose the value with the most occurrences in the parents (occurrence based scanning), or make a random, fitness proportional choice giving the highest chance to the fittest parent (fitness based scanning). In the tests reported here we always use uniform scanning.

Let us note that for simple bitstrings UPDATE is $x := x + 1$, i.e. we shift the markers one position to the right. With an easy modification scanning can be adapted for for order-based chromosomes. We only need to scan the genes of the $n > 1$ parents in such a way that no value is put in the child twice: UPDATE-ing is not $x := x + 1$, but

$$x := min\{y | y \geq x, gene(y) \text{ is not included in the child } \}.$$

It is easy to see that this scanning mechanism preserves the property of 'being a permutation'.

Parent 1:	1111010111000110
Parent 2:	1100010101000010
Parent 3:	0011101010101011
Parent 4:	0101010101100100
Child :	1111010111000010

Figure 2: Occurence based scanning crossover on bit patterns

4 Test functions

We have decided to perform experiments on numerical function optimization problems. We have chosen the DeJong functions F1, F2, F3, the Ackley, the Griewangk, the Michalewicz, the Rastigin and the Schwefel functions as test suite, [10, 13, 14, 20]. For the sake of completeness we present the function definitions we used.

The DeJong function F1 (also called the spherical function) is defined as:

$$f(\vec{x}) = \sum_{i=1}^{3} x_i^2,$$

where $-5.12 \leq x_i \leq 5.12$.

The DeJong function F2 (also called the Rosenbrock function) is defined as:

$$f(\vec{x}) = 100 \cdot (x_1^2 - x_2)^2 + (1 - x_1)^2,$$

where $-2.048 \leq x_i \leq 2.048$. The global minimum is zero at the point $(1, 1)$. The Rosenbrock function is characterized by an extremely deep valley along the parabola $x_1^2 = x_2$. The plots of these functions are given in Figure 3.

The DeJong function F3 is defined as:

$$f(\vec{x}) = \sum_{i=1}^{5} \text{integer}(x_i)$$

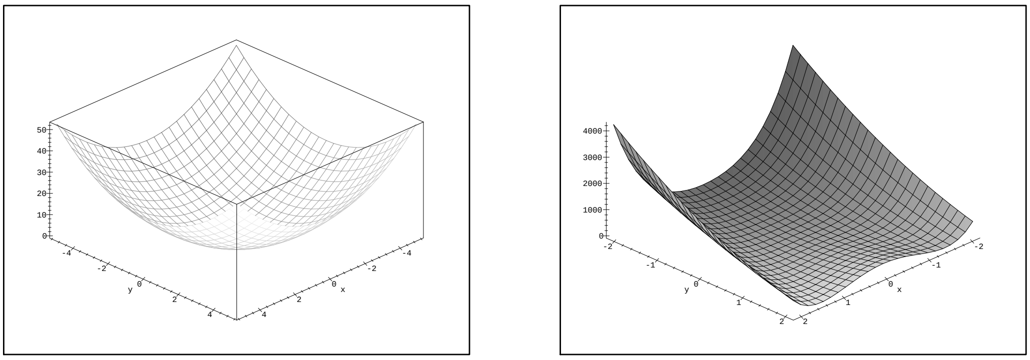

Figure 3: Plots of the spherical function (F1) and Rosenbrock's saddle (F2)

Where $-5.12 \leq x_i \leq 5.12$.

The Michalewicz function is a highly multimodal function to be maximized. It is defined as:

$$f(\vec{x}) = 21.5 + x_1 \cdot \sin(4\pi x_1) + x_2 \cdot \sin(20\pi x_2),$$

where $-3.0 \leq x_1 \leq 12.1$ and $4.1 \leq x_2 \leq 5.8$. The optimum of this function is 38.8503. The plots of these two functions are given in Figure 4.

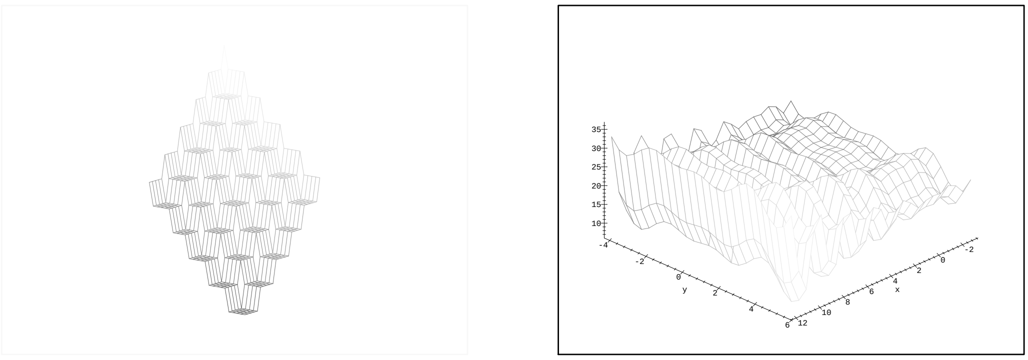

Figure 4: Plots of DeJong's F3 and the Michalewicz function

The Griewangk function is defined as:

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

where $n = 10$ and $-600 \leq x_i \leq 600$. The global minimum of zero is at the point $\vec{x} = (0, 0, 0, \cdots)$. This function has a product term introducing an interdependency between the

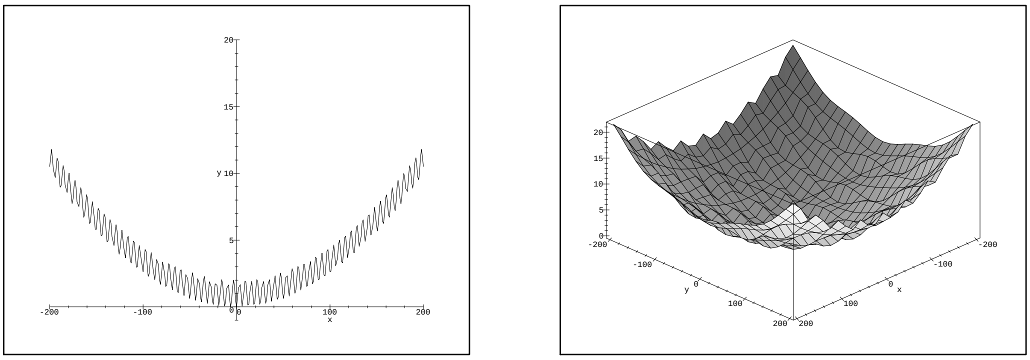

Figure 5: Plots of the Griewangk function for n=1 and n=2

variables. This is intended to disrupt optimization techniques that work on one function variable at a time. The plots is given in Figure 5.

The Rastrigin function is:

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

where $-5.12 \leq x_i \leq 5.12$. In our tests we used the value of $\alpha = 10.0$ and $n = 20$. The global minimum of zero is at the point $\vec{x} = (0, 0, \cdots)$. The primary characteristic of this function is the existence of many suboptimal peaks whose values increase as the distance from the global optimum point increases. The plot is given in Figure 6.

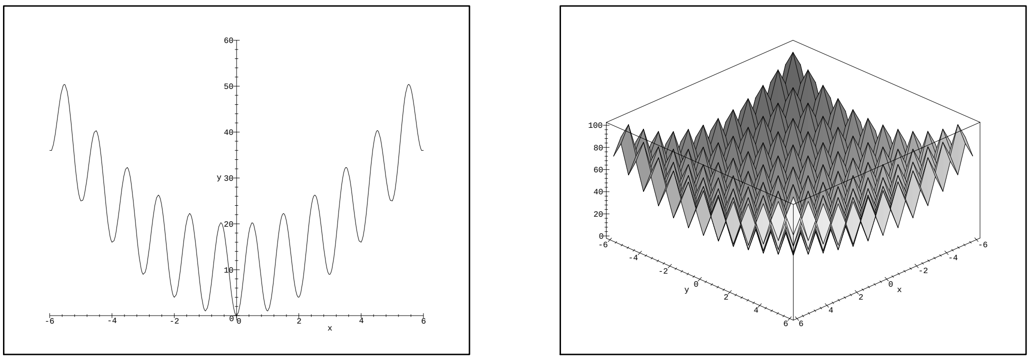

Figure 6: Plots of the Rastrigin function for n=1 and n=2

The Schwefel function is defined as:

$$f(\vec{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right)$$

where $-512.03 \leq x_i \leq 511.97$. In our tests we used the value $n = 10$. The global minimum of zero is at the point $\vec{x} = (420.9687, 420.9687, \cdots)$. The interesting characteristic of this function is the presence of a second-best minimum far away from the global minimum. The plot is given in Figure 7.

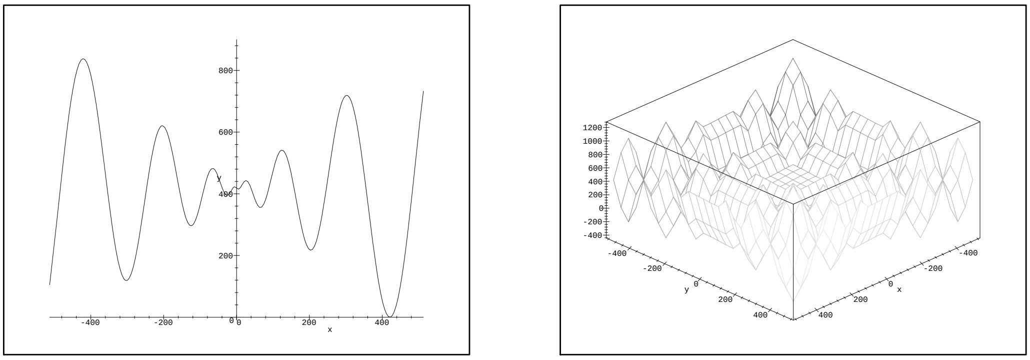

Figure 7: Plots of the Schwefel function for n=1 and n=2

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

The plot is given in Figure 8.

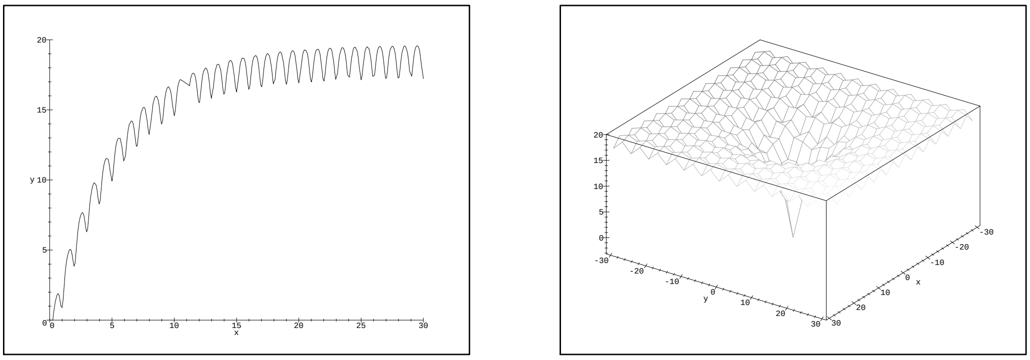

Figure 8: Plots of the Ackley function for n=1 and n=2

For each function we applied binary representation. The most important properties of the test functions and their representation are summarized in Table 1.

	F1	F2	F3	ACKL	GRIE	MICH	RAST	SCHE
dimension	3	2	5	30	10	2	20	10
chrom. length	30	22	50	600	200	33	400	210
global opt.	0	0	0	0	0	38.8503	0	0

Table 1: Properties of the test functions

5 GA setup

We compared the performance of different number of parents ranging from 2 to 15. Recall that for $n = 2$ diagonal crossover coincides with the usual 1-point crossover, thus we immediately obtained a comparison with a traditional GA as well. As discussed in Section 2, we compared diagonal crossover to 2-parent n-point crossover, in order to distinguish the effects of having more crossover points and those of using more parents.

In all of the experiments we used a modified version of the package LibGA [3] with the GA-setup exhibited in Table 2.

representation	fixed point binary w/o Gray coding
GA type	steady state
selection mechanism	ranked bias bias = 1.2
reduction mechanism	worst fitness deletion
nr. of parents	2-15
xover rate	0.7
mut. rate	1/chrom. length
pool size	200
max. nr. of evaluations	70.000
alternative termination condition	optimum hit or population converged
results averaged over	100 runs

Table 2: GA setup used in the experiments

6 Performance measures

When monitoring the performance we maintained different measures, namely

- success rate, that is the percentage of cases when an optimum was found;
- effectivity, i.e. quality of the solutions, measured by the best function value at termination, averaged over all runs;
- efficiency, i.e. speed, measured by the total number of function evaluations before termination, averaged over all runs.

Notice that success rate and effectivity are closely related, however, not the same. On the one hand, effectivity results are averaged over all runs. Hence, even with a high success rate, a few unlucky runs can corrupt the averages that measure effectivity. On the other hand, we work with a certain computational limit, the total number of fitness evaluations. This limit might be just too low to hit the optimum, but enough to approximate it closely. In other words, it can happen that success rates are low, but the average fitness values at the end of the evolution are close to the optimum. Therefore, the figures on effectivity provide extra information w.r.t. success rates. They demonstrate the robustness of the GA, that is they show how certain is the GA in approximating good solutions.

Note that using more parents can imply a higher number of function evaluations before termination. This, however, is not necessarily a negative effect if slower search comes together with higher success rates. In the meanwhile, hitting an optimum immediately terminates the search, thus using a more successful operator (more parents) can even reduce run times, thus yielding a double profit.

In the Appendices we display each performance measure for each test function and each number of parents.

7 Experimental results: diagonal crossover

We compare different number of parents in diagonal crossover, furthermore we compare diagonal crossover with n-point traditional crossover. Note that for n-point crossover the number of parents is always 2, for this operator the horizontal axis in the figures shows the number of chromosome segments (that equals the number of crossover points plus 1).

7.1 Success rate

Results on F1 and F3 were not too interesting: both diagonal crossover and n-point crossover found an optimum in 100 % of the cases for every n. The maximum number of fitness evaluations was obviously too high for these functions. This aspect will be further discussed in the section on efficiency. On the other test functions we observed that the success rates were increasing as n increased, for diagonal as well as n-point crossover. However, diagonal crossover steadily outperformed n-point crossover, see Figure 9, 10 and 11 in Appendix A.

7.2 Effectivity

The good success rate curves in the previous section make us expect that the graphs of the best function values show the same behaviour: improving if n is increased. On F1 and F3 the results were again 'boring', both operators reached the optimum in each run, resulting in a constant effectivity curve. Our observations on the other functions are summarized in Appendix B, in Figures 12, 13 and 14. The figures show a clear advantage of more parents on three functions and a somewhat zigzagging, but improving performance curve on the other ones.

7.3 Efficiency

As for efficiency we did not have a priori expectations on the behaviour of the GA. The details of the test results are given in Appendix C, in Figures 15, 16, 17 and 18. As the results show, on four out of the eight functions the total number of evaluations roughly decreases with increasing n. Thus, here we have a double advantage of more parents: the search becomes better (see success rates) and also faster. On the Michalewicz, the Rastrigin and the Schwefel functions this is not the case. Recall that the DeJong's F1 and F3 wre not depicted w.r.t. the other performance measure. The reason is that they were easy, success rates were 100%, hence the effectivity curves were also constant, suggesting that no advantage is gained by using more parents. The curves on efficiency, however, clearly show that more parents are better here too: the search is accelerated.

8 Experimental results: scanning crossover

In Appendices D - F we display each performance measure for each test function and each number of parents for the scanning crossover. For comparison between scanning and diagonal crossover we include the results for diagonal crossover in the figures. Note that the number of parents in the tests for scanning crossover was 1-10.

8.1 Success rate

For the scanning crossover we also observed that the success rates were increasing as n increased, see Figures 19, 20 and 21. However, the correlation between n and the success rates was less monotonous than for diagonal crossover. Nevertheless, as the plots show, the optimal number of parents was higher than 2, with the Schwefel function as the only exception. The clearest advantage of high n's can be seen on the Griewangk function.

8.2 Effectivity

Our observations are summarized in Figures 22, 23 and 24 in Appendix E. Looking at the effectivity curve belonging to the Schwefel function we can see the advantage of more parents that would have remained invisible if we had considered success rates only.

8.3 Efficiency

The Figures 25, 26 and 27 in Appendix F exhibit the efficiency curves. On the Griewangk, Ackley and Rastrigin functions we can observe an accelerated search. This, together with the success rate results indicates a double profit as discussed in Section 7.3.

9 Conclusions and future work

From a strict optimization point of view the success rate is the most interesting performance measure. It is also the basis of calculating the expected amount of processing needed to solve a problem with a given probability, cf. [12] chapter 8. Table 3 shows the optimal versions of the genetic operators and the corresponding success rates for each test function. Within brackets we displayed the success rate of the 2-parent versions.

test	Scanning Xover		Diagonal Xover		N-point Xover	
function	#par	succ.	#par	succ.	#Xover points	succ.
F2	7	.91 (.73)	11	.88 (.38)	11	.84
Mic	10	.72 (.57)	15	.76 (.34)	15	.6
Schw	2	.015 (.015)	15	.24 (.00)	10	.1
Grie	10	.48 (.22)	14	.32 (.04)	10	.15
Ras	5	.10 (.00)	13	.28 (.00)	15	.06
Ackl	8	.90 (.84)	15	.89 (.00)	10	.24

Table 3: Optimal nr. of parents and corresponding success rates (within brackets the results for 2 parents)

It appears immediately that the optimal number of parents is always higher than 2 with one exception, the Schwefel function. (But, as discussed in Section 8.2, even on this function more parents have a clear advantage in terms of better approximation of the optimum, see Figure 24.) The gains achieved by using more than two parents are substantial, especially for the diagonal crossover. The figures within brackets show an interesting phenomenon. Namely, on all tests functions the standard uniform crossover performs much better than 1-point crossover (diagonal crossover for two parents). Looking at the results of diagonal crossover and n-point (2 parent) crossover we can see that the better performance of diagonal crossover is not only the consequence of applying more crossover points, but the higher number of parents contributes considerably.

In the last ten years Evolutionary Computation has grown to a large and diverse field covering Genetic Algorithms [4, 10, 13], Evolution Strategies [1, 15], Evolutionary Programming [9] and Genetic Programming [12].

Parallel to, and partly caused by, the increasing variety of Evolutionary Algorithms (EAs) the usefulness of sexual recombination in GAs has been questioned. For instance, Eshelman and Schaffer [8] look for 'crossover's niche', i.e. problems where pair-wise mating has competitive advantages. They do find such problems, in the meanwhile suspect that sexual recombination in GAs might be less powerful than generally believed. Hordijk and Manderick [11] investigate the usefulness of recombination on (epistatic) NK landscapes. Evolutionary Programming uses unary operators only. Evolution Strategies are traditionally based on mutation as main search operator [2], but use recombination too. Interestingly, the so-called global recombination in ES allows that a child receives its genes from more than two parents. However, for each bit in the child only two parents are considered, the increased parent number comes by randomly choosing two new parents for each bit. In the comparisons reported in [2] ES and EP exhibit better performance than GAs.

One possible way to improve GA performance is to incorporate new features in the GA, i.e. features that do not belong to the traditional GAs paradigm. Some recent attempts that were successful use Lamarckian/Baldwinian effects, or a problem decomposition, [14, 20]. By applying multi-parent recombination operators we follow another approach. We do remain in the original GA paradigm and boost GA performance by raising the extent of sexuality by allowing 'orgies', i.e. multi-parent reproduction. The results on our test suite consisting of difficult functions showed that this approach is very fruitful on function optimization. Note, however, that our goal was not to make GAs superior function optimizers. For optimizing (continuous) functions Evolution Strategies or other evolutionary techniques using hill-climbing or local optimization might be a better alternative.

Our ongoing and future research goes in two directions. On the one hand more research is needed to understand this new phenomenon of multi-parent reproduction. Theoretical analysis of schema survival and schema combination rates, [16], can illuminate the success of using $n > 2$ parents. Looking at positional and distributional bias, [7], in diagonal, respectively scanning crossover can explain their differences and determine their right domain of application. Experiments on carefully designed test functions, such as Royal Road, deceptive functions, NK landscapes can help to determine 'multi-parent's niche', i.e. to identify those problems where multi-parent operators have advantage over classical ones. An interesting sub-issue is the question of the optimum number of parents.

Current research is directed to enhancement of the multi-parent recombination operators, in order reduce the random noise and get a more reliable behavior. On the other hand, our experiments put the issues of disruptivity and selection pressure in the focus, and especially the interaction between these two. Bit variance and co-variance [17], early commitment behaviour [8], the right balance between exploration and exploitation are heavily influenced by these two issues. Multi-parent recombination allows for fine tuning the disruption rate. In combination with appropriate selection pressure this can be a powerful tool in designing high performance GAs.

Acknowledgements

For the tests we used a modified version of LibGA, [3]. The plots of the test functions were made by MAPLE, the performance curves were created by gnuplot.

10 Appendices

Appendix A: Success rates of diagonal crossover

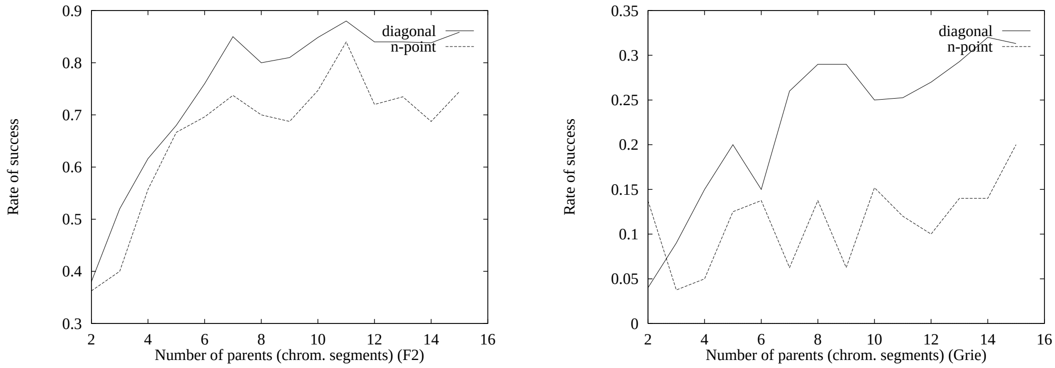

Figure 9: Diagonal crossover: success rates on F2 and the Griewangk function

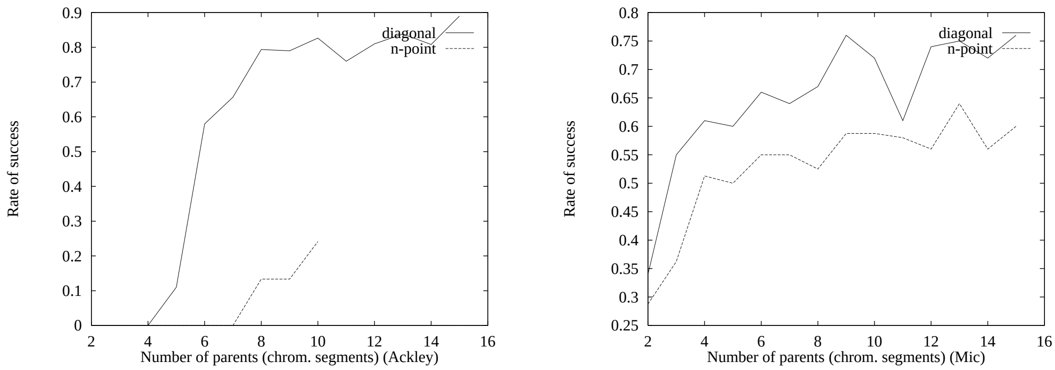

Figure 10: Diagonal crossover: success rates on the Ackley and the Michalewicz functions

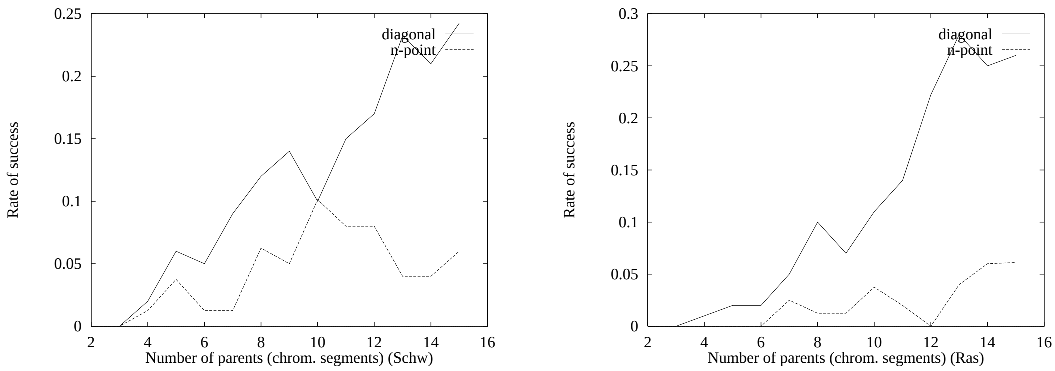

Figure 11: Diagonal crossover: success rates on the Schwefel and the Rastrigin functions

Appendix B: Effectivity of diagonal crossover

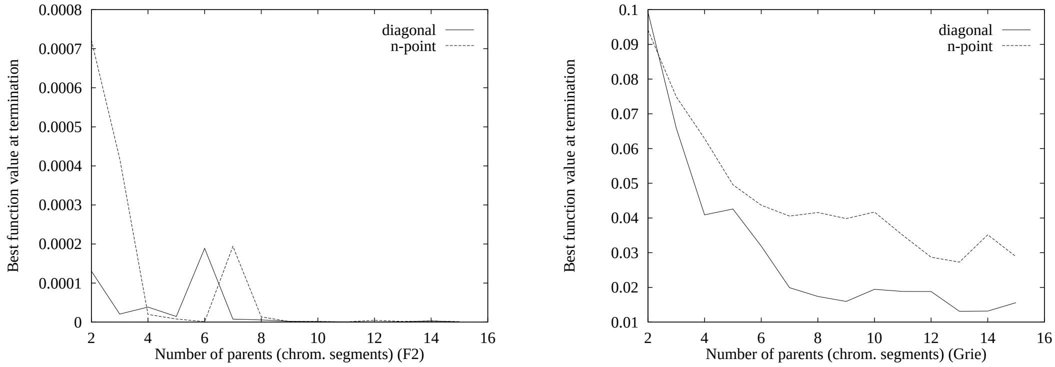

Figure 12: Diagonal crossover: effectivity on F2 and the Griewangk function

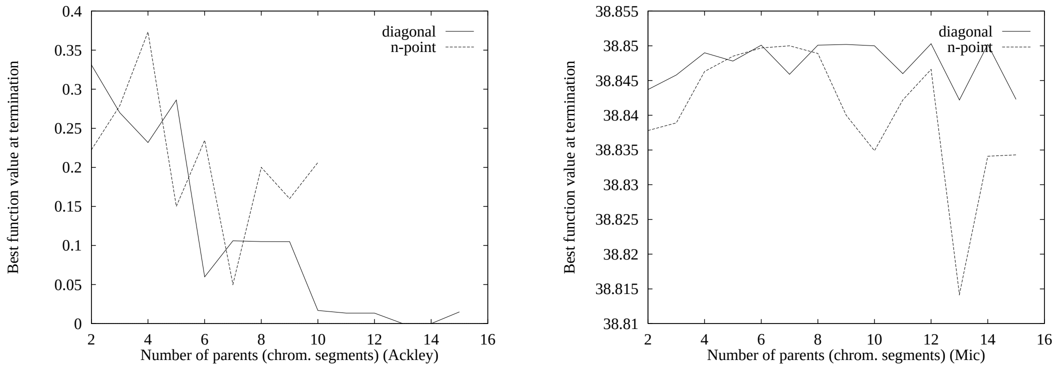

Figure 13: Diagonal crossover: effectivity on the Ackley and the Michalewicz functions (the latter one is a maximization problem)

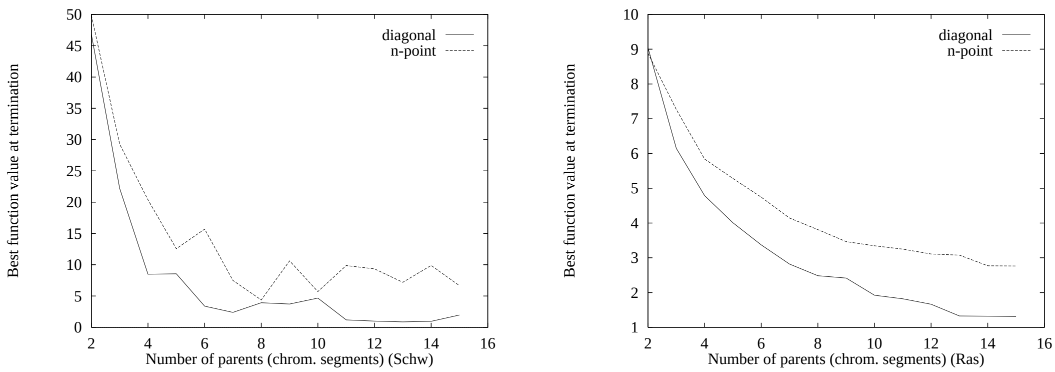

Figure 14: Diagonal crossover: effectivity on the Schwefel and the Rastrigin functions

Appendix C: Efficiency of diagonal crossover

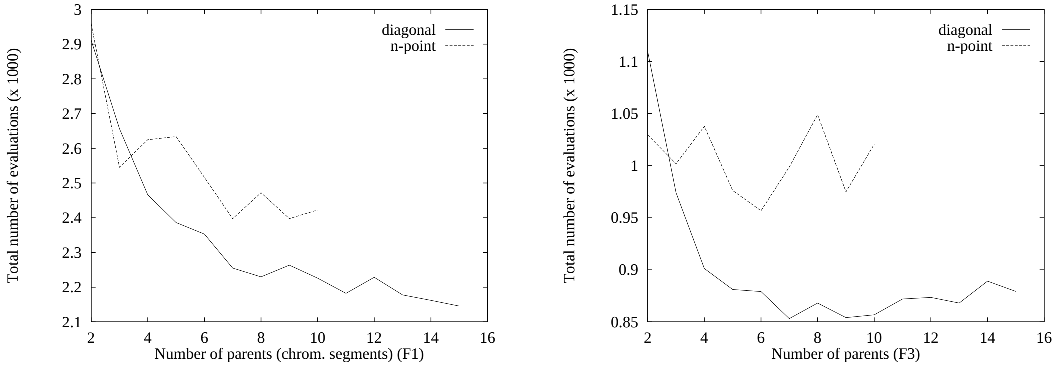

Figure 15: Diagonal crossover: efficiency on F1 and F3

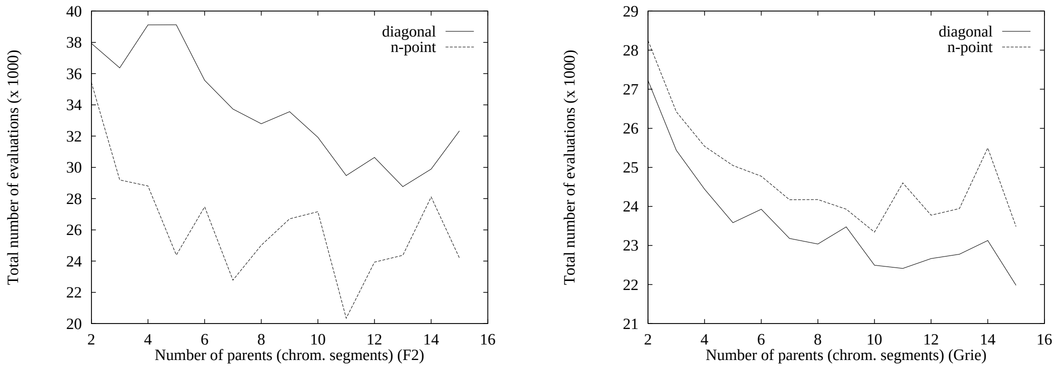

Figure 16: Diagonal crossover: efficiency on F2 and the Griewangk function

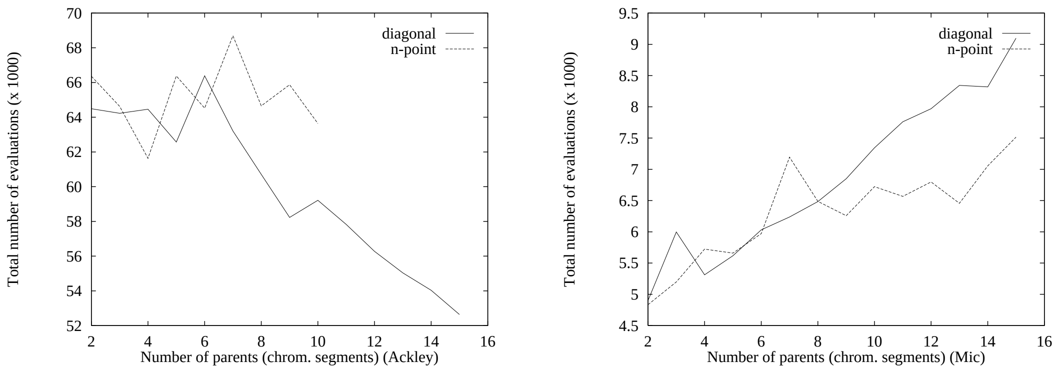

Figure 17: Diagonal crossover: efficiency on the Ackley and the Michalewicz functions

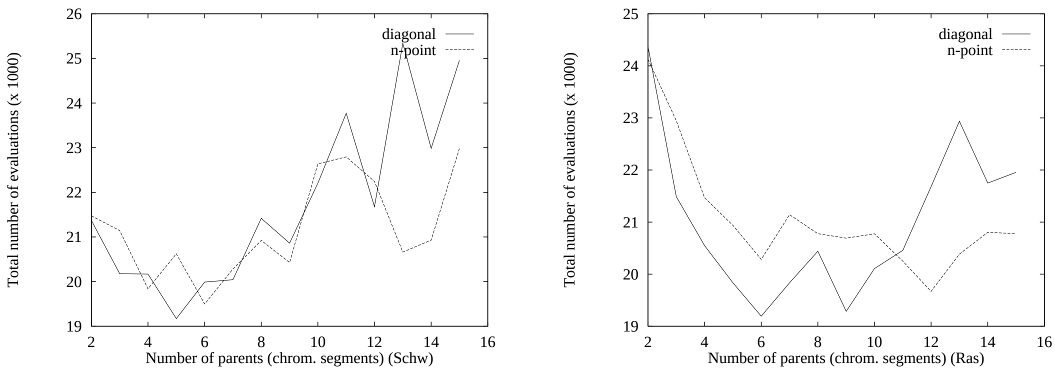

Figure 18: Diagonal crossover: efficiency on the Schwefel and the Rastrigin functions

Appendix D: Success rates of scanning crossover

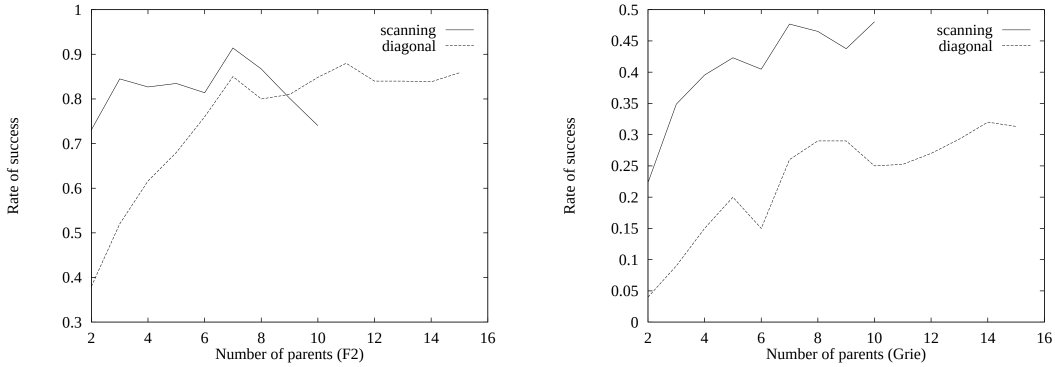

Figure 19: Scanning crossover: success rates on F2 and the Griewangk function

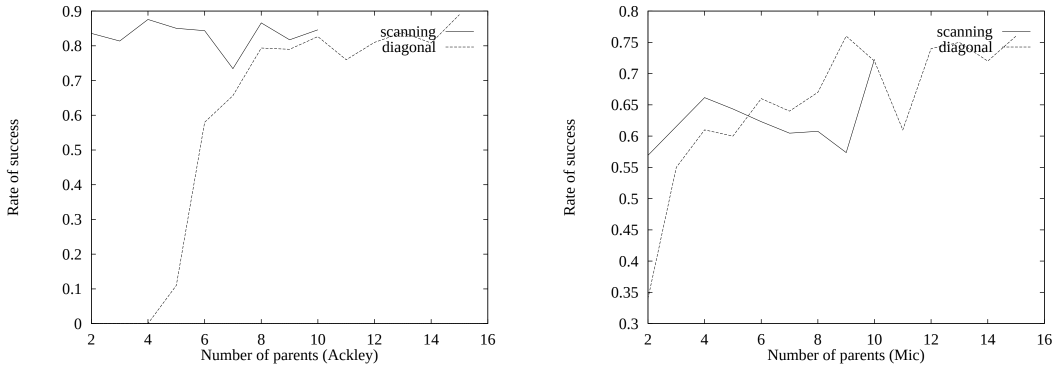

Figure 20: Scanning crossover: success rates on the Ackley and the Michalewicz functions

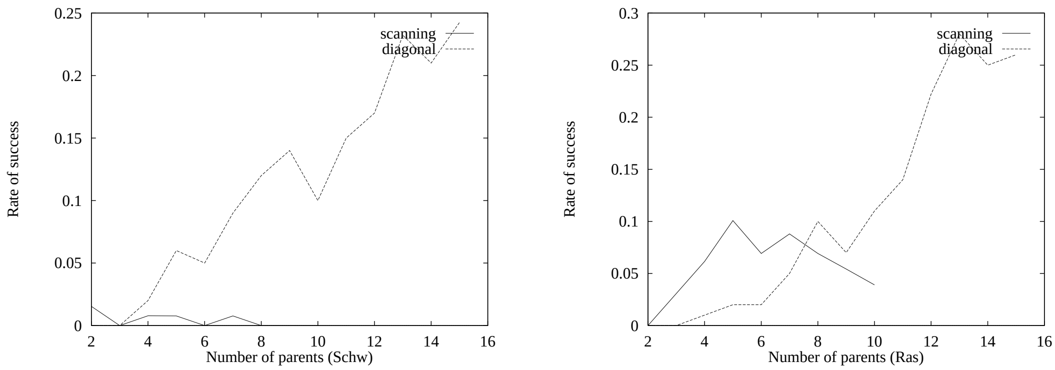

Figure 21: Scanning crossover: success rates on the Schwefel and the Rastrigin functions

Appendix E: Effectivity of scanning crossover

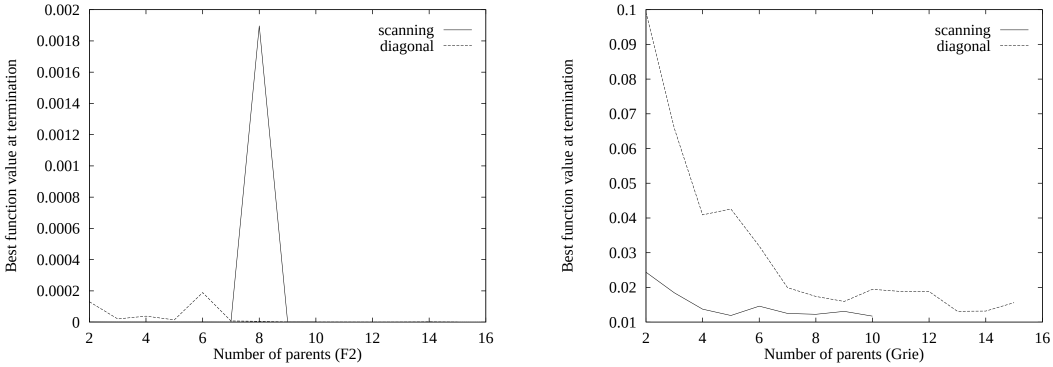

Figure 22: Scanning crossover: effectivity on F2 and the Griewangk function

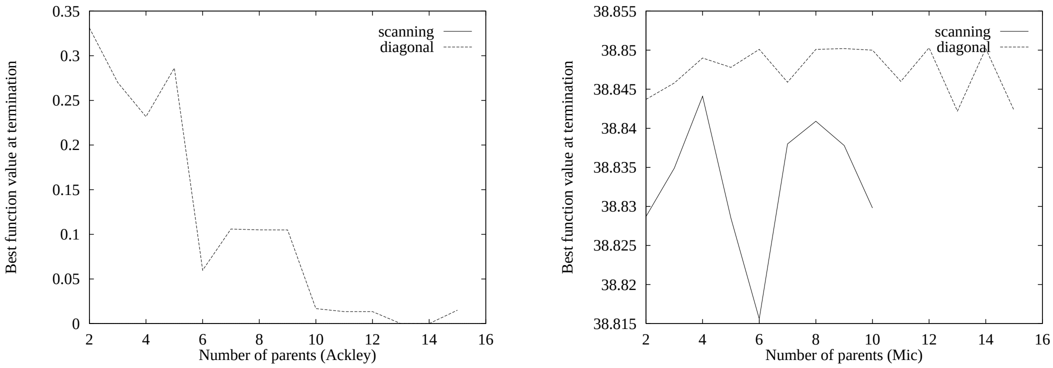

Figure 23: Scanning crossover: effectivity on the Ackley and the Michalewicz functions (the latter one is a maximization problem)

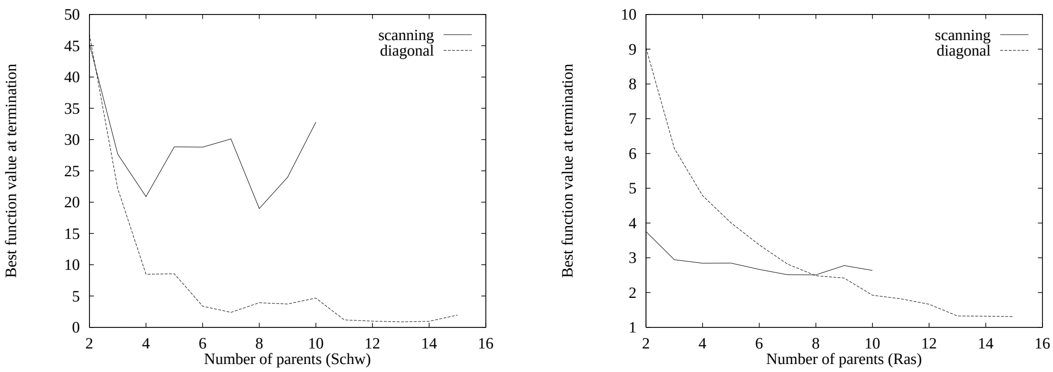

Figure 24: Scanning crossover: effectivity on the Schwefel and the Rastrigin functions

Appendix F: Efficiency of scanning crossover

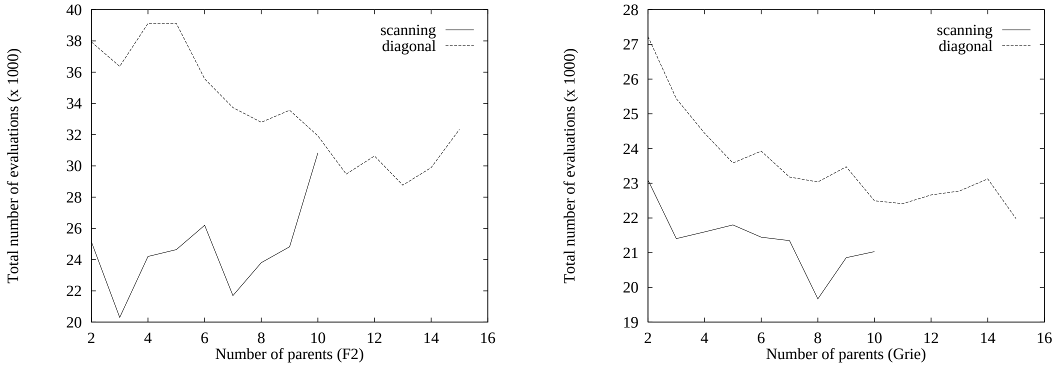

Figure 25: Scanning crossover: efficiency on F2 and the Griewangk function

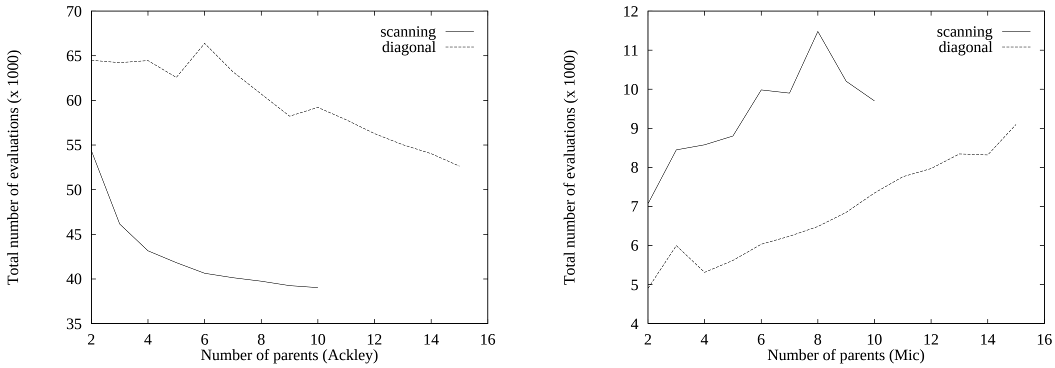

Figure 26: Scanning crossover: efficiency on the Ackley and the Michalewicz functions

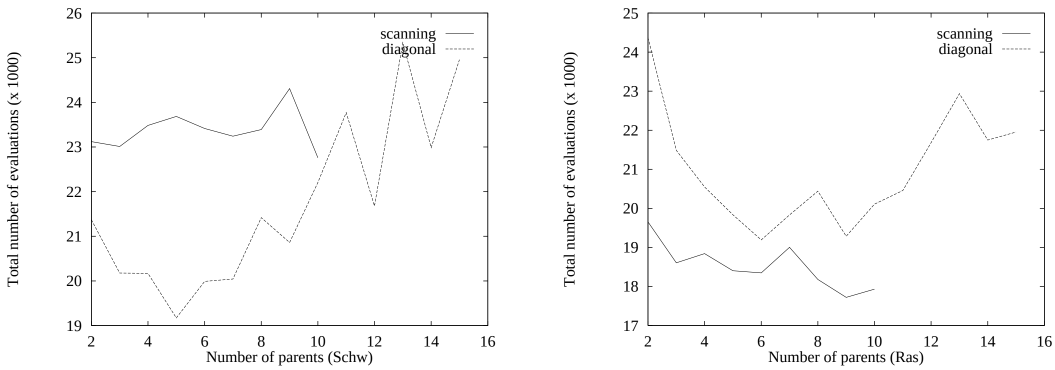

Figure 27: Scanning crossover: efficiency on the Schwefel and the Rastrigin functions

References

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice, Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, 1995.

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

[3] A.L. Corcoran and R.L. Wainwright. LibGA: A user-friendly workbench for order-based genetic algorithm research. In H. Berghel E. Deaton, K.M. George and G. Hedrick, editors, *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, pages 111–118. ACM Press, 1993.

[4] L. Chambers (editor). *Practical Handbook of Genetic Algorithms.* CRC Press, 1995.

[5] A.E. Eiben, P-E. Raué, and Zs. Ruttkay. Genetic algorithms with multi-parent recombination. In *Parallel Problem Solving from Nature - 3, LNCS 866*, pages 78–87, 1994.

[6] 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, LNAI 929*, pages 934–945. Springer-Verlag, 1995.

[7] L.J. Eshelman and R.A. Caruana andJ.D. Schaffer. Biases in the crossover landscape. In *Third International Conference on Genetic Algorithms*, pages 10–19, 1989.

[8] L.J. Eshelman and J.D. Schaffer. Crossover's niche. In *Fifth International Conference on Genetic Algorithms*, pages 9–14, 1993.

[9] D.B. Fogel. *Evolutionary Computation.* IEEE Press, 1995.

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

[11] W. Hordijk and B. Manderick. The usefulness of recombination. In *Third European Conference on Artificial Life, LNAI 929*, pages 908–919. Springer-Verlag, 1995.

[12] J.R. Koza. *Genetic Programming.* MIT Press, 1992.

[13] Z. Michalewicz. *Genetic Algorithms + Data structures = Evolution programs.* Springer-Verlag, second edition, 1994.

[14] M.A. Potter and K.A. De Jong. Cooperative coevolutionary approach to function optimization,. In *Parallel Problem Solving from Nature - 3, LNCS 866*, pages 249–257. Springer-Verlag, 1994.

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

[16] G. Syswerda. Uniform crossover in genetic algorithms. In *Third International Conference on Genetic Algorithms*, pages 2–9, 1989.

[17] D. Thierens. *Analysis and Design of Genetic Algorithms.* PhD Thesis, Universiteit Leuven, 1995.

[18] 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*, pages 137–146, 1995.

[19] 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*, 1995.

[20] D. Whitley, V. Scott Gordon, and K. Mathias. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature - 3, LNCS 866*, pages 6–15. Springer-Verlag, 1994.