Genetic algorithms with multi-parent recombination

Conference Paper · October 1994

DOI: 10.1007/3-540-58484-6_252 · Source: DBLP

3 authors, including:

A. E. Eiben
Vrije Universiteit Amsterdam
336 PUBLICATIONS **11,935** CITATIONS

SEE PROFILE

Zsófia Ruttkay
Moholy-Nagy University of Art and Design, Budapest
134 PUBLICATIONS **1,813** CITATIONS

SEE PROFILE

Genetic algorithms with multi-parent recombination

A.E. Eiben, P-E. Raué, Zs. Ruttkay

Artificial Intelligence Group
Dept. of Mathematics and Computer Science
Vrije Universiteit Amsterdam
De Boelelaan 1081a
1081HV Amsterdam
e-mail: {gusz, peraue, zsofi}@cs.vu.nl

Abstract We investigate genetic algorithms where more than two parents are involved in the recombination operation. We introduce two multi-parent recombination mechanisms: gene scanning and diagonal crossover that generalize uniform, respecively n-point crossovers. In this paper we concentrate on the gene scanning mechanism and we perform extensive tests to observe the effect of different numbers of parents on the performance of the GA. We consider different problem types, such as numerical optimization, constrained optimization (TSP) and constraint satisfaction (graph coloring). The experiments show that 2-parent recombination is inferior on the classical DeJong functions. For the other problems the results are not conclusive, in some cases 2 parents are optimal, while in some others more parents are better.

1 Introduction

In nature, new individuals are always created through either asexual (one parent) or sexual (two parents) reproduction. People hardly ever wonder why the number of parents is restricted to one or two. For a mathematician or computer scientist, however, this restriction should not be self-evident. Still, almost no GA publications report on recombination mechanisms based on more than two parents, we are aware of only two, namely [Müh89, Ser92], and in the context of Evolution Strategies that of [Bäc93]. Here we examine whether multi-parent recombination should be avoided for practical reasons or it offers advantages and is just overlooked.

Aside from this motivation we have another argument for investigating multi-parent recombination. Namely, one would expect that by biasing the recombination operator the performance of the GA would improve. A problem indepedent way to incorporate a bias into the recombination operator can be based on selecting n parents and applying some limited statistical analysis on the allele distribution within the parents. A very slight bias is introduced when choosing an allele from the parents uniform randomly. Looking at the number of occurrences of a certain allele at a position and choosing the most common allele introduces a stronger bias. Inheriting the number of alleles proportional to the fitness-value of the parents is a more sophisticated, but still problem-independent mechanism. In the next section we give detailed descriptions of a number of multi-parent recombination operators. In section 3 we present test results on 6 different problems. An evaluation of the test results is presented in section 4. Open issues and future work are discussed in section 5.

2 Multi-parent genetic operators

A multi-parent operator uses two or more parents to generate children. In this section

we define two multi-parent recombination mechanisms: gene scanning and diagonal crossover that generalize uniform, respecively n-point crossovers. In this paper we concentrate on the gene scanning mechanism, we describe and investigate five versions of scanning that differ in the type of bias applied when creating children.

2.1 Gene scanning techniques

Gene scanning, as introduced in [Eib91], is a general mechanism that produces one child from n > 1 parents in the following way. First, a number of markers are assigned, one for each of the parents and one for the child. The child-marker traverses all the positions in the child from left to right, one at a time. For each position, the markers for the parents are updated so that when choosing a value for the gene currently marked in the child, the parent markers indicate the choices. This algorithm is shown in pseudo-code in figure 2.1.

```
1  Initialize parent markers
2
3  FOR child_marker = 1 TO CHROMOSOME_LENGTH DO
4      update parent markers
5
6      child.allele[child_marker] = choose from values indicated by
                                    the parent markers
7  END_FOR
```

Figure 2.1. The general gene scanning mechanism

The two characterizing features of all gene scanning procedures are the (parent) marker update mechanism (line 4) and the mechanism that chooses the value to be inherited by the child (line 6). By changing this latter mechanism we have defined three specific scanning techniques, namely: uniform, occurrence-based and fitness-based scanning. The marker update mechanism is discussed at the end of this section.

Uniform scanning The classical uniform crossover traverses two parents and the two children from left to right, and for each position in child 1 chooses randomly whether to inherit from parent 1 or parent 2 (the second child is created by reversing the decisions). U-Scan uses a mechanism with is basically the same, the difference is that only one child is created and instead of only 2 parents any number of parents can be used. Each parent has an equal chance of contributing an allele to be inherited by the child, which means that the probability to inherit an allele from parent i, P(i), and the expected number of genes to inherit from parent i, E(i), are defined as follows:

$$P(i) := \frac{1}{number\ of\ parents},\ E(i) := P(i) \times CHROMOSOME\_LENGTH$$

Occurrence-based scanning Occurrence-based scanning (OB-Scan) is based on the premise that the value which occurs most often in a particular position in the parents (which are selected based on their fitness) is probably the best possible value to choose. Choosing one of the marked values is thus done by applying a majority function. If no value occurs more than any other one (the majority function is undecided), then the value which is encountered first is chosen in our implementation Another way of doing this would be to break ties randomly, proportionally to the number of occurrences. Note, that the distrubution used should not be uniform, since it that case this randomized occurrence-based scanning would yield uniform scanning.

Fitness-based scanning Fitness-based scanning (FB-Scan) uses roulette wheel selection when deciding from which parent an allele will be inherited. This means that if parent i has a fitness value of f(i) the probability that a value for a position in the child is inherited from this parent is:

$$P(i) := \frac{f(i)}{\sum f(i)}$$

Using this mechanism the child will inherit more alleles from fitter parents, since the number of alleles inherited from parent i is $E(i) = P(i) \times CHROMOSOME\_LENGTH$.

Adapting scanning to different representation types It is possible to define scanning procedures for different representation types by changing the marker update mechanism. For representations where all positions are independent (no epistasis), the marker update mechanism is simple, the parent markers are all initialized to the first position in each of the parents, and each step the markers are all increased by one (thus traversing the parents from left to right). Problems where epistasis occurs may need a more sophisticated marker update mechanism. In particular, for other order-based representation one needs a marker update mechanism which ensures that no value is added to the child twice. This can be obtained by increasing the marker in each parent until it marks a value which has not been added to the child yet. An example of how this marker update mechanism works is shown in figure 2.2.

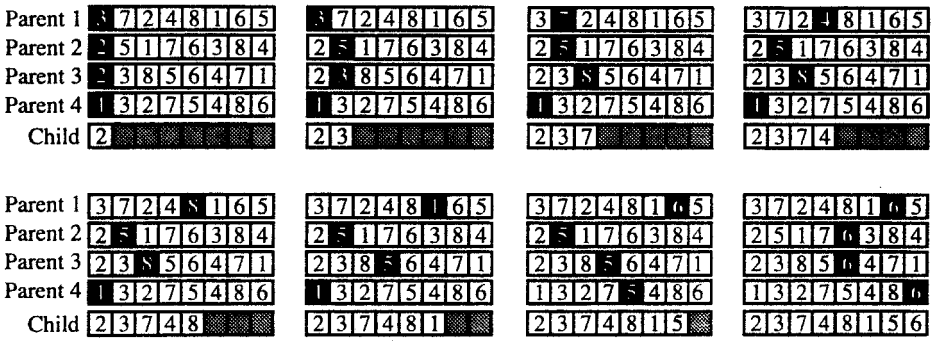


Figure 2.2. OB-Scan on order-based representation

2.2 Adjacency-based crossover

The adjacency-based crossover (ABC) is a special case of scanning, designed for order-based representations where relative positioning of values is important, e.g. the TSP. The main differences between the scanning procedures described above and ABC are the way in which the first value is selected and the marker update mechanism. The first gene value in the child is always inherited from the first gene value in the first parent. The marker update mechanism is as follows: for each parent its marker is set to the first successor of the previously selected value which does not occur in the child yet. (For this mechanism each individual is seen as a cycle).

A crossover similar to ABC was proposed in [Whi91]. Whitley's crossover differs from ABC in the number of parents applied (Whitley uses 2 parents) and the choice of a value to inherit when all the immediate successors to a city have already been

inherited by the child. For example, suppose that the successors to city D are cities A, E, F and H (in parents 1, 2, 3 and 4 respectively), and furthermore that these four cities have already been incorporated into the child. Whitley's crossover would randomly choose a city not already included in the child. ABC will check the successors of city A in parent 1, city E in parent 2, city F in parent 3 and city H in parent 4 (and if any of these have already been included in the child, their successors), the city to be added to the child will be chosen from among these successors.
We define two types of ABC, namely occurrence-based (OB-ABC) and fitness-based (FB-ABC) similarily to occurrence-based and fitness-based scanning. The difference is the marker update mechanism. OB-ABC is shown in figure 2.3.

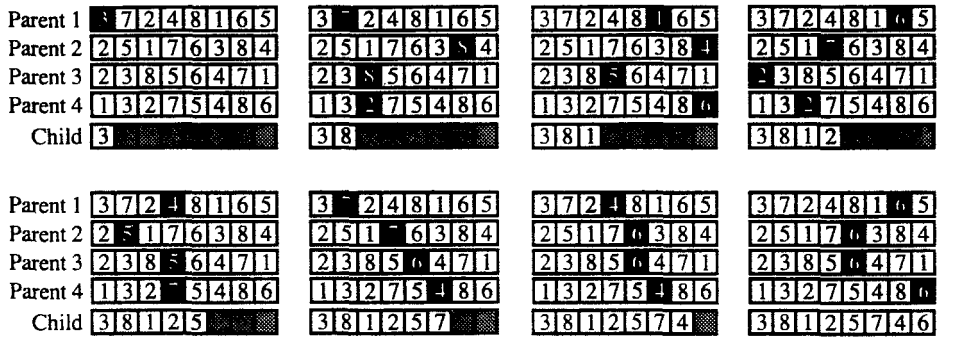


Figure 2.3. OB-ABC

2.3 Diagonal multi-parent crossover

Diagonal crossover is a generalization of n-point crossover, which creates 2 children from 2 parents. Diagonal crossover uses n ≥ 1 crossover points and creates n+1 children from n+1 parents as the following figure illustrates.

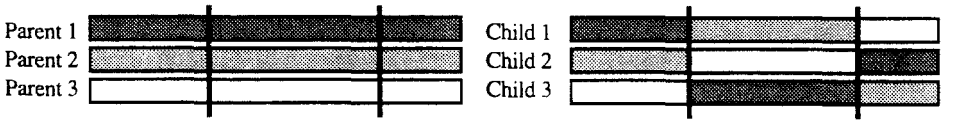


Figure 2.4: The diagonal crossover operator for 3 parents.

As mentioned before, in this paper we concentrate on scanning. We will report on the effect of the number of parents for diagonal crossover in forthcoming publications

3 The experiments

We decided to study multi-parent recombination within the framework of scanning by testing and comparing the three basic versions of scanning (uniform, occurrence-based and fitness-based) together with the scanning-like ABC which is tailored to routing problems. We were mainly interested in the following issues:

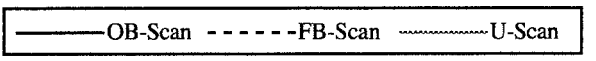
- Is scanning better than classical crossover?
- Which version of scanning is the best?
- What is the optimal number of parents?
- Does a relation exist between the type of the problem and
  - the optimal version of scanning (as a refinement of the second question),
  - the optimal number of parents (as a refinement of the third question)?

We have tested five numerical functions to be optimized and two 'real' problems. For numerical optimization we have chosen four well-known DeJong functions (the formulas were taken from [Gol89]) and a function from [Mic92] because these functions provide a test-bench with variously shaped functions. The first 'real' problem is the TSP because it is well known constrained optimization problem in the GA community and is important in practice. We use an instance concerning 30 cities from [Oli87] to keep running times down. For the second 'real' problem we have chosen graph 3-coloring: the nodes of a graph must be colored using three colors so that no neighboring nodes have the same color. This is known to be a very tough constraint satisfaction problem. (Note that this problem is different from the graph coloring problem discussed in [DavL91].) We considered 'difficult, but solvable' graphs: graphs which can be colored with 3 colors, but for which solutions are difficult to construct [Che91]. In other papers we report on heuristic GAs that outperformed the applicable classical deterministic search methods, [Eib94a, Eib94b].

For the DeJong functions and the function from [Mic92] we applied the standard binary representation. For the TSP and graph coloring we used order-based representation, furthermore we tested graph coloring with a representation based on a ternary alphabet. As for the number of parents we tested every multi-parent version of our crossovers for any number of parents from 2 to 10. The results shown here were obtained by running the GA 100 or 200 times and calculating performance averages for each number of parents. We used the generational GA model with 100% crossover rate and 50% mutation rate (per individual, not per position). The maximum number of generations was set to 500 (with earlier termination if the optimum was reached).

3.1 The test results

We tested the performance of uniform, occurrence-based and fitness-based scanning on the DeJong functions F1, F2, F3 and F4 (all minimization problems). In section 4 we compare the results with classical 1-point crossover. (The legend is the same for each graph.)



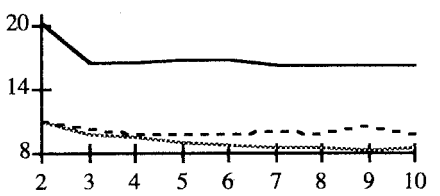


Figure 3.1. The average nr. of generations needed to find the optimum for OB/FB/U-Scanning on F1 (chrom. length = 30)

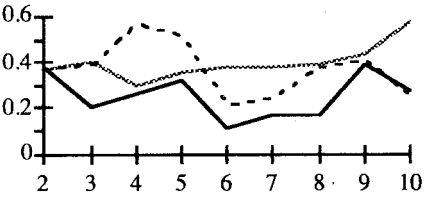


Figure 3.2. The best f-value after 500 generations for OB/FB/U-Scanning on F2 (chrom. length = 22)

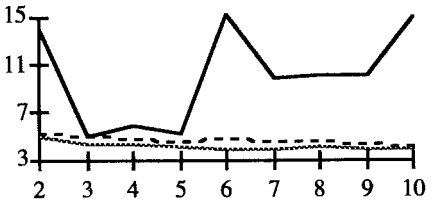


Figure 3.3. The average nr. of generations needed to find the optimum for OB/FB/U-Scanning on F3 (chrom. length = 50)

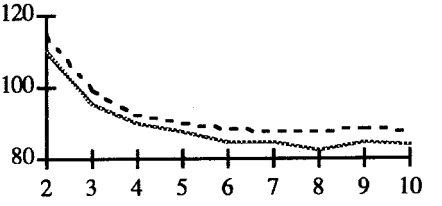


Figure 3.4a. The average nr. of generations needed to find the optimum for FB/U-Scanning on F4 (chrom. length = 240)

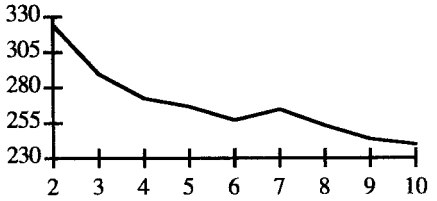


Figure 3.4b. The average nr. of generations needed to find the optimum for OB-Scanning on F4 (chrom. length = 240)

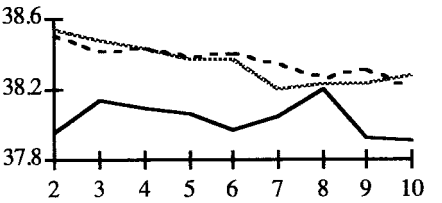


Figure 3.5. The best f-value after 500 generations for OB/FB/U-Scanning on the function from [Mic92], pg. 34 (maximization problem, chrom. length = 33)

For the graph coloring problem we applied primarily order-based representation, decoding permutations similar to the way in which it is done for scheduling problems, [Eib94b]. We tested two graphs with 90 nodes and 5 edges per node. This is known to be approximately the toughest topology for graph coloring.

	number of parents								
	2	3	4	5	6	7	8	9	10
OB-Scan	32	10	18	14	4	6	6	2	4
FB-Scan	40.5	34	28	27	28.5	21	29	26	28
U-Scan	46	29	34	36.5	34.5	36	28.5	32.5	33
OB-ABC	36	22	8	10	6	8	4	16	4
FB-ABC	20	16	10	18	20	16	24	14	20

Figure 3.6. Percentage of cases when a solution was found for graph coloring with 90 nodes (order based representation, chrom. length = 90, population size = 180)

Besides order-based representation of colorings we also tested a representation with a ternary alphabet, each symbol denoting one color. In this case we looked at the performance of the 'standard' scanning mechanisms.

	number of parents								
	2	3	4	5	6	7	8	9	10
OB-Scan	31	2.5	4.5	2	5.5	1.5	1	0	1.5
FB-Scan	35	26	34	34.5	33	32.5	33.5	31.5	29
U-Scan	42.5	35	32.5	28.5	27	26	32.5	30	27

Figure 3.7. Percentage of cases when a solution was found for graph coloring with 30 nodes (ternary alphabet, chrom. length = 30, population size = 100)

When testing the TSP we found that the best 'traditional' order-based crossover was order1, which we used to compare our results against. For TSP only occurrence-based scanning and ABC had an optimal number of parents higher than two, the other scanning techniques performed best with two parents, and only FB-ABC was able to outperform the order1 crossover. The results are summarized in the following figure showing the effectivity of the GA.

	number of parents								
	2	3	4	5	6	7	8	9	10
OB-Scan	514.2	500.3	501.7	506.6	500.6	504.5	504.6	504.8	500.0
FB-Scan	492.6	492.6	493.4	493.7	497.3	499.9	499.0	497.3	500.7
U-Scan	491.6	492.2	500.3	494.5	493.4	497.7	498.3	495.1	500.7
OB-ABC	509.1	486.6	484.8	477.4	475.6	478	474.9	472.5	476.6
FB-ABC	453.2	458.7	457.9	459.5	461.7	457.3	462.8	463.6	461.6
order1	461.6								

Figure 3.8. The length of the best route after 500 generations (order based representation, chrom. length = 30, population size =100).

4 Evaluation of the tests

In this part we review the test results grouped around three main questions: advantages of scanning with respect to classical recombination mechanisms, different types of bias in scanning and the effect of different numbers of parents.

4.1 Scanning and classical recombination

Considering the five numerical optimization problems we have tested, we observed that in 4 out of the 5 cases (fitness-based) scanning outperformed 1-point crossover. On the DeJong functions it outperformed 1-point crossover by respectively, 42%, 19%, 40% and 59%. On the function from [Mic92] fitness-based scanning was 1.3% worse than 1-point crossover. Let us remark that on the function F2 OB-Scan proved to be better than FB-Scan and outperformed 1-point crossover by as much as 70%.

For the TSP we first tested the classical order based crossovers order1, order2, position, cycle, PMX and uox (provided by our test environment LibGA, [Cor93]) and observed that order1 outperformed the rest. Then we ran tests to compare different versions of scanning with order1. Here, the fitness-based version of ABC (a multi-parent operator designed specifically for the TSP) turned out to be the best scanning mechanism, and was slightly (1.8%) better than order1.

4.2 Different types of bias in scanning

On 4 out of the 5 numerical optimization problems fitness-based scanning was superior to the uniform and occurrence-based versions. The exception was F2, where OB-Scan was better, even though on the other 4 functions it was the worst of the three. In section 5 we give a possible explanation for this. For graph coloring we found that choosing genes uniform randomly proved to be better than choosing fitness- or occurrence-based. On the TSP, performance for the uniform and fitness-based scanning was approximately the same, OB-Scan was inferior to both. As for the adjacency-based crossover operators, FB-ABC clearly outperformed OB-ABC.

4.3 The effect of different number of parents

The results on the DeJong functions show that the performance increases as the number of parents is raised above 2. The tests, however, provide no real insight into the optimal number of parents. For instance, the optimal number of parents for fitness-based scanning was respectively, 9, 4, 6-10, 10 for F1, F2, F3 and F4. We also observed that the different versions of scanning show different behaviour, i.e. reach optimal performance at different numbers of parents. Nevertheless, on all of the DeJong functions we tested the optimal number was higher than 2. For the function from [Mic92] 2 parents proved to be optimal, although the difference in performance between different numbers of parents remained below 1%. This suggests that for this function the number of parents does not really matter.

The TSP shows a somewhat similar picture. We have tested 5 multi-parent recombination mechanisms and each of them has shown little variance in performance for different numbers of parents. For three of them the results became worse (by at most 2.3%) when the number of parents was raised above 2; for the two occurrence-based recombination mechanisms the optimal number of parents was 10, respectively 9, but the gain with respect to 2 parents did not exceed 7.2%. For graph coloring the data clearly shows that 2 parents are superior.

When increasing the number of parents extra costs are introduced. These costs occur due to the extra time needed to select additional parents and the time needed to identify the gene to be inherited. Gene inheritance costs are almost negligible on binary strings. In the worst case (e.g. order-based representation), the increase can be quadratic, for instance for OB-Scan. However, even in this case, the decrease in the number of generations may compensate the time increase per generation.

5 Open questions and future work

In spite of the 23.000 runs in total, we are aware of the limitations of the present investigation. Thus, instead of general conclusions we discuss some questions that can only be answered reliably if more knowledge is collected on the phenomenon of multi-parent reproduction.

5.1 When is the optimal number of parents larger than 2?

Using a greater number of parents, the child creation is biased based on a larger sample from the search space. If a function has many (almost) optimal points separated by 'gaps' of poor points then sampling might not help. The different good elements in a population may very well belong to the 'gravity field' of different

solutions. Hence, the larger number of parents does not provide information about the *same* solution, and the generated child will most likely be somewhere in-between the (different) gravity fields of the parents, instead of getting closer to one of them. Hence, one might not expect improvement by increasing the number of parents for 'hedgehog-like' functions. This was justified for FB-Scan and U-Scan on the function taken from [Mic92], and we expect similar results for the F5 DeJong function.

If there are relatively few (optimal) solutions, OB-Scan is expected to be the most effective (as was shown for F2), since in this case the fit parents close to different solutions differ on many bits, and the occurrence-based inheritance of genes is likely to produce children close to that solution which is close to many of the parents.

Obviously, the number of parents which can be used in the recombination mechanism is limited by the number of individuals in the pool. The question is whether increasing the number of parents always improves the performance of the GA. Our experiments support two kinds of conclusions. In some cases, increasing the number of parents improves the performance, though the degree of improvement becomes smaller and smaller. This was observed for FB-Scan and U-Scan for F3, and roughly for all multi-parent crossovers for F1 and F4. In other cases, the performance improves up to a certain number of parents and decreases, or oscillates afterwards. This was observed for the OB-Scan for F2 and F3.

The largest improvement when increasing the number of parents was achieved in the case of OB-Scan, independently from the problem. This is not surprising, as a larger sample provides more reliable information on strong gene patterns, and the two-parent OB-Scan in our implementation always reproduced the first parent.

In most of the cases we experienced a much larger improvement when the number of parents was changed from 2 to 3 than in the successive steps. We do not have a sound explanation for this phenomenon, though from the practical point of view it would be very useful to know that it is not worth while to go beyond 3-4 parents.

Another interesting question is whether the type of (the bias in) a crossover is the decisive factor with respect to the performance. In other words, if a 2-parent crossover is better than another crossover, does the same apply for the n-parent version for any $n > 2$ as well? (For the optimization problems we tested this was approximately true, with the exception of F2.)

5.2 Other issues

In our first experiments, we were interested whether the increase in the number of parents results in better performance. The optimal number of parents obviously depends on the length of the individuals and on the alphabet, or in general, on the size of the search space (in addition to its topology). A further investigation could aim at deciding the optimal number of parents in different stages of the GA, especially, in the case of premature convergence. By introducing incest prevention the performance of the multi-parent crossovers may improve.

Another aspect which needs to be investigated is the number of children generated from a given number of parents. So far, we have only studied n-parents-1-child versions of scanning-like crossovers, whereas more children can be created easily.

Looking at the performance as we have done here is the most common way in which to compare different genetic operators or genetic algorithms. Nevertheless, to get a better understanding of this novel phenomenon of multi-parent recombination we need a 'higher resolution' picture of the behaviour of our GAs. This can be obtained by a theoretical schema survival analysis, or monitoring extra features along the evolution, such as:

- uniformity within the population measured by the average Hamming distance of randomly chosen pairs of individuals;
- power of sexual reproduction measured by the child fitness - parent fitness ratio.

Acknowledgements
We wish to thank A.L. Corcoran who provided us the LibGA library that we used for our experiments.

References

[Bäc93] T.-B. Bäck and H.-P. Schwefel, An Overview of Evolutionary Algorithms for parameter Optimization, Journal of Evolutionary Computation 1(1), 1993, pp. 1-23.

[Che91] P. Cheeseman, B. Kenefsky and W.M. Taylor, Where the really hard problems are, Proc. of IJCAI-91, 1991, pp. 331-337.

[Cor93] A.L. Corcoran and R.L. Wainwright, LibGA: A User Friendly Workbench for Order-Based Genetic Algorithm Research, Proc. of Applied Computing: Sates of the Art and Practice, 1993, pp. 111-117.

[DavY91] Y. Davidor, Epistasis Variance: A View-point on GA-Hardness, Proc. of FOGA-90, 1991, pp. 23-35.

[DavL91] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[Eib91] A.E. Eiben, A Method for Designing Decision Support Systems for Operational Planning, PhD Thesis, Eindhoven University of Technology, 1991.

[Eib94a] A.E. Eiben, P.-E. Raué and Zs. Ruttkay, Solving Constraint Satisfaction Problems Using Genetic Algorithms, proceedings of the 1st IEEE World Conference on Computational Intelligence, to appear in 1994.

[Eib94b] A.E. Eiben, P-E., Raué, Zs. Ruttkay, GA-easy and GA-hard Constraint Satisfaction Problems, Proc. of the ECAI'94 Workshop on Constraint Processing, LNCS Series, Springer-Verlag, to appear in August, 1994.

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

[Gre85] J. Grefenstette, R. Gopal, B. Rosmaita and D. Van Gucht, Genetic Algorithms for the Travelling Salesman Problem, Proc. of ICGA-85, 1985, pp.160-168.

[Mic92] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1992.

[Müh89] H. Mühlenbein, Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization, Proc. of ICGA-89, 1989, pp. 416-421.

[Oli87] I.M. Oliver, D.J. Smith and J.R.C. Holland, A study of permutation crossover operators on the travelling salesman problem, Proc. of ICGA-87, 1987, pp. 224-230.

[Ser92] G. Seront and H. Bersini, In Search of a Good Evolution-Optimization Crossover, Proc. of PPSN 2, 1992, pp. 479-488.

[Whi91] D. Whitley, T. Starkweather and D. Shaner, The traveling salesman and sequence scheduling: quality solutions using genetic edge recombination, In: [DavL91], pp. 350-372.