Competing Crossovers in an Adaptive GA Framework *

A.E. Eiben, I.G. Sprinkhuizen-Kuyper, B.A. Thijssen
Department of Computer Science
Leiden University
{gusz,kuyper}@wi.leidenuniv.nl

Abstract

In this paper we report the results of experiments on multi-parent reproduction in an adaptive genetic algorithm framework. An adaptive mechanism based on competing subpopulations is incorporated into the algorithm in order to detect the best crossovers. Experiments on a number of test functions designed for studying crossover performance show that multi-parent reproduction is superior to traditional two-parent crossover, but the adaptive mechanism is not able to reward better crossovers according to their performance. Nevertheless, the adaptive algorithm exhibits comparable performance to the non-adaptive variant using the best crossover alone. This implies that it is sound and safe to use an adaptive GA with competing subpopulations/crossovers, instead of performing time consuming comparisons in search of the best operators.

1 Introduction

The present investigation combines research goals and techniques from two sub-areas in evolutionary computation. One area is the study of reproduction operators. There has been a lot of research on the power of different crossover operators and on the power of crossover (binary reproduction operator) versus mutation (unary reproduction operator) [9, 10, 11, 17, 16]. Recently, n-ary crossovers were introduced and shown to have increased performance for higher arities on some numerical optimization problems and

*Technical Report 97-11, Department of Computer Science, Leiden University. Available as ftp://ftp.wi.leidenuniv.nl/pub/CS/TechnicalReports/1997/tr97-11.ps.gz

NK-landscapes [6, 7, 8]. Here we investigate fitness landscapes that were specifically designed for studying crossovers in [9, 16, 17]. Adaptivity and emergent properties in simulated evolutionary processes form the second area this research is based upon. Having several options for a certain GA parameter or component implies that a choice has to be made between them. Choosing the best crossover operator usually happens by testing them each and finally using the one that exhibits the best performance. Alternatively, using adaptive or self-adaptive mechanisms, [12], allows the GA making the choice. Simultaneously using more operators and adaptively modifying their application rate, [4, 14, 22], or encoding usage information in the chromosomes and applying self-adaptation, [20], have been tried in the past.

In this paper we compare crossovers in an adaptive GA framework based on competing subpopulations. In nature, different species sharing the same environment may develop different strategies to ensure survival and, most preferably, to increase their 'share of the cake'. In this paper we experiment with an evolutionary system where different subpopulations (species) search by different strategies. Individuals of different subpopulations are not allowed to mate, interference between the species is restricted to migration. The adaptive population redistribution mechanism is designed in such a way that subpopulations with successful strategies grow and those of unsuccessful strategies shrink. In this study we restrict ourselves to using different crossover operators in each subpopulation, thus restricting the meaning of a 'strategy' to 'crossover'. Our experiments are carried out in a Genetic Algorithm framework and have a three-fold objective:

- investigating whether multi-parent crossovers are better than others on the chosen landscapes,
- investigating whether a GA simultaneously using more crossovers and adaptively tuning on the applied crossovers is better than a traditional GA,
- investigating whether the built-in self-organization mechanism detects the differences in performance, that is whether the subpopulations of successful crossovers become larger than other ones.

The rest of the paper is organized as follows. In Section 2 we briefly review and illustrate the crossovers used in this investigation. In Section 3 the self-organization mechanism is described. Thereafter in Section 4 we present the experimental setup and summarize the test results. Finally, we draw conclusions and sketch further research in Section 5.

2 Crossover operators

The traditional N-point crossover [5] uses two parents and creates two children. It selects N crossover points and the children are built by alternatingly taking segments from the first, respectively second parent. Diagonal crossover [7] uses N parents and selects $(N-1)$ crossover points in the bitstrings. The N children are created by combining the segments from the parents in a diagonal manner; the i-th child starts with the first segment of the i-th parent and each next segment will be taken from the following parent, wrapping around after the last parent.

For both types of crossovers (N-point and diagonal) we also test a variant which only creates one child (the first one). Illustrations for 2-point crossover and 3-parents diagonal crossover are given in Figure 1 and Figure 2.

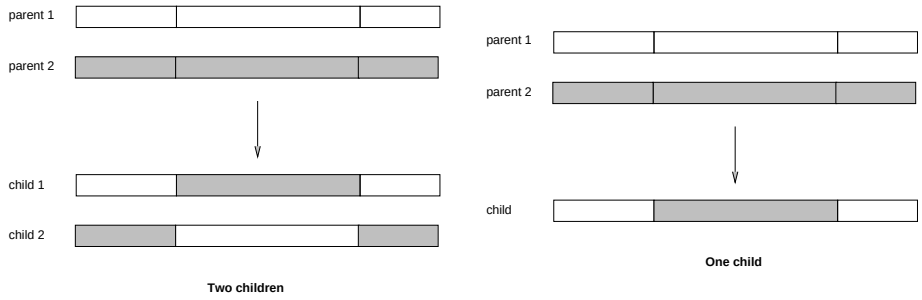

Figure 1: 2-Point crossover with two children (left) and one child (right)

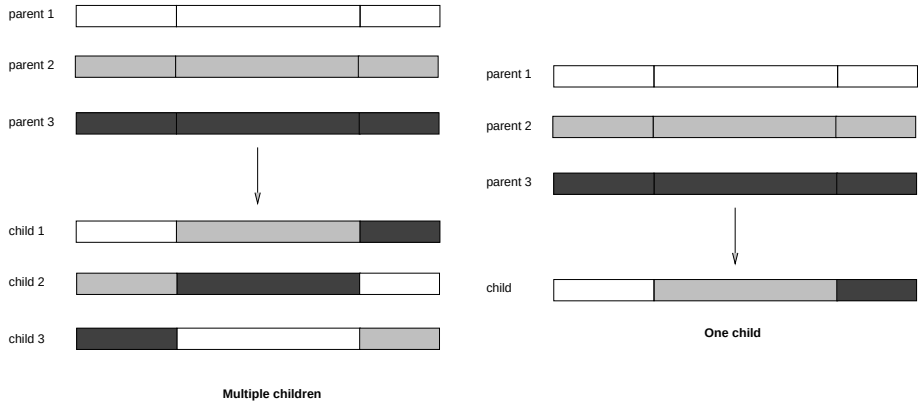

Figure 2: 3-Parent diagonal crossover with three children (left) and one child (right)

3 Migration and redivision mechanisms

Redistribution of the subpopulations is done by two mechanisms: *migration* is to increase the size of successful subpopulations, while *redivision* (applied less frequently) serves the opposite goal, it is to give a second chance to inferior subpopulations and make them increase in size.

There are many different mechanisms possible to handle migration between the different subpopulations. Competing subpopulations each using a different search strategy has been used in [2, 15, 18, 19], for instance. The mechanism we use takes a number of randomly selected individuals from each subpopulation and puts them in a migration pool. In this phase inferior subpopulations have to deliver more individuals. After that the individuals in the migration pool will be randomly redistributed over the subpopulations in such a way that each subpopulation receives approximately equal number of individuals. The exact definition is given below.

Let F_i be the average fitness of subpopulation i, F_{min} and F_{max} respectively the average fitness of the worst and best subpopulation (fitness to be maximized), $\Delta F_i = F_{max} - F_i$. If $F_{max} = F_{min}$ then no migration takes place. Otherwise the normalized $\Delta F_i{}^*$ values can be calculated as follows:

$$\Delta F_i{}^* = \frac{\Delta F_i}{F_{max} - F_{min}}$$

Each subpopulation P_i has to deliver $c \times \Delta F_i{}^*$ of its individuals to the migration pool, where $0 \leq c \leq 1$ is a parameter of the mechanism. The i-th migration-subpool MP_i will be thus formed by randomly choosing $c \times \Delta F_i{}^* \times |P_i|$ individuals from P_i, rounded to a natural number, and the migration pool will be $MP = \cup_i MP_i$. The individuals from the migration pool are distributed over the subpopulations uniform randomly, so if there are M subpopulations, then each subpopulation receives $\lfloor \frac{|MP|}{M} \rfloor$ individuals. The individuals which are still in the migration pool will be assigned to the populations with the highest average fitness in descending order, one each. This mechanism is illustrated in Figure 3.

Besides migration we also use a redivision mechanism to give the smallest subpopulations another chance. This prevents a crossover that is inferior in the beginning, but would be successful later on, from disappearing by receiving new members from the redivision pool. The redividing mechanism takes a percentage of every subpopulation, where the smaller subpopulations give fewer individuals than the bigger ones, and redivides them equally over all subpopulations. This will bring the sizes of the subpopulations closer together. The redivision of the subpopulations is done in two steps. First, $d \times |P_i|$ (rounded to a natural number) individuals will be randomly taken

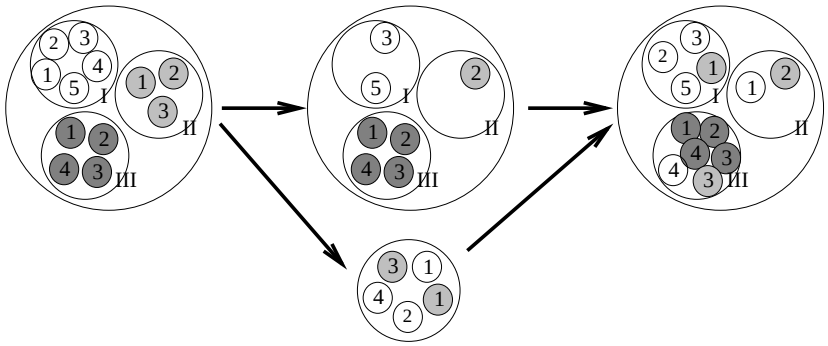

Figure 3: Illustration of the migration mechanism: subpopulations I, II and III before migration (left), reduced subpopulations and migration pool (middle) and subpopulations after migration (right).

from each subpopulation and put in the redivision pool. The parameter d indicates what percentage of the subpopulation P_i should be given away. The second step is to redivide the individuals from the redivision pool back to the subpopulations. This will be done uniformly and randomly, so if there are M subpopulations, then every subpopulation P_i will receive $\lfloor \frac{|RP|}{M} \rfloor$ individuals randomly. The individuals which are still in the redivision pool will be assigned to the populations with the highest average fitness in descending order, one each.

The individuals moving by both mechanisms adopt the crossover operator assigned to their new subpopulation.

4 Experiments

Seven different fitness landscapes were used for this research, Onemax [17], Twin Peaks [9], Plateau [17], Plateau-d [17], Trap [17], Trap-d [17] and Royal Road [16]. These functions were adjusted a little, for instance they were reversed to maximization problems, because our GA maximizes the objective function, see [21] for technical details. For all these functions we used a bitstring length of 100, the optimal function values are summarized in Table 1.

In the tests we used a Steady State GA for each subpopulation and 10 subpopulations, initially containing 50 individuals each. Tournament selection with tournament size 4 and worst fitness deletion strategy, crossover rate $p_c = 0.7$ and mutation rate $p_m = 0.005$ is used everywhere. The constants for the migration and the redivision mechanism are set to $c = 0.75$, and

Problem	maximum fitness
Onemax	100
Twin Peaks	100
Plateau	100
Plateau-d	100
Trap	200
Trap-d	200
Royal Road	600

Table 1: Maximum fitness values of the applied test functions

$d = 0.25$[1]. We ran three series of experiments which differ in the definition of one generation, or one cycle, in the Steady-State GA the subpopulations use. In each series we executed 50 independent runs on each test function, the presented results are averaged over the 50 runs.

4.1 Test series A: one crossover per generation

In the first series we used one crossover operation in each subpopulation before inserting the offspring in the population. An overview of crossovers used in different subpopulations is given in Table 2.

Subpopulation	0	1	2	3	4	5	6	7	8	9
Points in N-point	1	2	3	4	5	-	-	-	-	-
Parents in diag.	-	-	-	-	-	2	3	4	5	6
Children per op.	2	2	2	2	2	2	3	4	5	6

Table 2: Test series A: crossovers used in different subpopulations

The maximum number of evaluations is 60000, migration is applied after every 30 evaluations, the frequency of applying the redividing mechanism is set to 7500 evaluations. On each test function we observed that the crossovers with more parents were assigned larger subpopulations by the adaptive mechanism. We present the figures on the seven functions in Figure 4. In these, and the following figures the x axis represents the elapsed time, measured by the number of fitness evaluations, graded by the number of population

[1]These values were chosen after a few preliminary tests which are not reported here.

updates. On the y axis we depict the 10 subpopulations, while the z axis shows the size of the subpopulation.

These outcomes suggest that the crossovers with more parents are better, therefore the adaptive mechanism increases the corresponding subpopulations. Nevertheless, the results might be an artifact of the different number of children created by different operators. Namely, diagonal crossover with arity k creates k children, while the N-point operators create only 2. This implies that subpopulations using diagonal crossover with higher arity perform more search in one selection-reproduction-replacement cycle. If we call the cardinality of the offspring population in the Steady State GA the *generational gap*, (which terminology is somewhat different from that used in generational GAs), then we can say that the generational gap is bigger for subpopulations using higher arity crossover operators.

4.2 Equal generational gaps

To filter out the possible effect of different generational gaps, we performed two new series of experiments where this gap was kept equal for each subpopulation. To this end we tested two realizations. The first one is based on the one child versions of the operators, cf. Section 2. Since one application of any crossover operator now creates the same number of children we can use the same mechanism as before updating the subpopulations (i.e. inserting the offspring) after performing one crossover. Thus, this first option is based on modifying the operators and leaving the update mechanisms in tact. The second realization of the equal generational gaps is based on the complementary idea. We use the same operators as in the first test series, but update the subpopulations after different number of crossover applications, in such a way that the number of offspring created before updating equals for each subpopulation. In the following experiments the maximum number of evaluations is set to 50000.

4.2.1 Test series B: one child per crossover

In this test series we replaced diagonal crossover for two parents (which is the same as 1-point crossover) with the 7 parents version. The assignment of crossovers to subpopulations is given in Table 3.

The migration frequency is 100 evaluations, the redividing frequency is 5000 evaluations. We present the curves for the seven functions in Figure 5.

Subpopulation	0	1	2	3	4	5	6	7	8	9
Points in N-point	1	2	3	4	5	-	-	-	-	-
Parents in diag.	-	-	-	-	-	3	4	5	6	7
Children per op.	1	1	1	1	1	1	1	1	1	1

Table 3: Test series B: crossovers used in different subpopulations

4.2.2 Test series C: more children per crossover

In these tests we set the number of created offspring before replacement in the Steady State GA each subpopulation worked with to 24. Accordingly we modified the number of parents for diagonal crossover as shown in Table 4. The migration frequency was 240 evaluations, the redividing frequency was 5000 evaluations. The results for the seven functions are presented in Figure 6.

Subpopulation	0	1	2	3	4	5	6	7	8	9
Points in N-point	1	2	3	5	7	-	-	-	-	-
Parents in diag.	-	-	-	-	-	3	4	6	8	12
Children per op.	2	2	2	2	2	3	4	6	8	12

Table 4: Test series C: crossovers used in different subpopulations

4.2.3 Comparing the results of the test series

The outcomes of the test series B and C differ from those of series A. The adaptive mechanism only slightly, if at all, rewards the higher arity operators. There are (at least) two different explanations for this fact. The first possible reason can be that diagonal crossover with more parents is not better than traditional 2-parent crossovers on the test suite used in this study. The second one is that adaptivity based on competing subpopulations as implemented here is not able to detect and reward better crossovers. To this end we ran some cross-checking experiments with 2-parent 1-point crossover and 6-parent diagonal crossover on the whole test suite.[2]

[2]Testing all N's used in the experiments would cause an explosion in the number of tests, therefore not performed.

4.3 1-Point vs. 6-parent diagonal crossover

For the cross-checking experiments we used only one subpopulation of 500 individuals and executed 50 independent runs on each test function. We compared the two crossovers according to three measures, the mean best function values (MBF), the Average number of Evaluations to Solution (AES) with their standard deviations and the Success Rates (SR), that is the percentage of all runs when an optimal solution was found. The results of the control experiment on series B (one child per crossover) are given in Table 5, those of cross-checking series C (number of children according to the original operator definitions) are shown in Table 6.

	6-parents diagonal crossover					1-point crossover				
Problem	MBF	sd	AES	sd	SR	MBF	sd	AES	sd	SR
Onemax	100.0	0.0	3095	253.1	100	100.0	0.0	5691	595.8	100
Twin Peaks	100.0	0.0	3839	521.2	100	100.0	0.0	6021	596.9	100
Plateau	100.0	0.0	8060	8127.5	100	97.6	3.2	25021	9105.0	60
Plateau-d	99.2	1.8	23479	10337.4	84	93.9	4.9	35063	5368.7	26
Trap	189.8	5.1	3701	102.3	6	168.3	8.7	-	-	0
Trap-d	138.8	7.2	-	-	0	136.7	6.8	-	-	0
Royal Road	595.8	29.4	10228	9369.2	92	480.2	112.9	30050	6970.3	46

Table 5: Diagonal crossover with 6 parents vs. 1-point crossover, one undivided population, one child per crossover

	6-parents diagonal crossover					1-point crossover				
Problem	MBF	sd	AES	sd	SR	MBF	sd	AES	sd	SR
Onemax	100.0	0.0	3155	261.1	100	100.0	0.0	5813	522.5	100
Twin Peaks	100.0	0.0	4012	632.1	100	100.0	0.0	6012	564.8	100
Plateau	99.9	0.7	9012	9791.8	98	98.0	2.6	29817	9535.2	62
Plateau-d	99.3	1.7	22279	9680.3	86	94.5	5.1	37705	5171.9	34
Trap	189.3	6.3	4360	75.4	10	169.7	7.5	-	-	0
Trap-d	139.7	9.0	-	-	0	135.9	7.9	-	-	0
Royal Road	583.2	57.0	12920	11113.0	92	477.2	122.8	23434	6868.1	48

Table 6: Diagonal crossover with 6 parents vs. 1-point crossover, one undivided population, original operator variants

These tables clearly show that diagonal crossover with 6 parents outperforms the traditional 2-parent 1-point crossover on each test function. On

the Onemax and the Twin Peaks function the MBF and SR results are identical, but the diagonal crossover is significantly faster (lower AES). Besides, the standard deviations of diagonal crossover are lower. It is interesting to compare the one child variants and the original versions of both operators. The differences are not significant, indicating that this aspect is not critical for the performance of the operators. For this reason we only compare the adaptive GA using the original operator variants with the non-adaptive GAs. Table 7 presents the analogue of Table 6 for the adaptive GA using the crossover-subpopulation assignment as given in Table 4.

	Multiple Subpopulations (setup C)				
Problem	MBF	sd	AES	sd	SR
Onemax	100.0	0.0	4940	325.9	100
Twin Peaks	100.0	0.0	6359	974.4	100
Plateau	100.0	0.0	7855	6350.3	100
Plateau-d	97.9	2.5	29315	8009.3	58
Trap	191.4	5.3	6913	190.0	10
Trap-d	142.9	8.5	-	-	0
Royal Road	579.0	63.0	12332	7793.3	90

Table 7: Performance of the adaptive GA using the original operator variants

From this table it is clear that the adaptive GA outperforms the standard GA using 1-point crossover and has comparable performance with the GA using diagonal crossover with 6 parents (see Table 6).

5 Conclusions and future work

As discussed in the Introduction, we had a three-fold objective with this investigation. As for our first question concerning the differences in the successfulness of crossovers, we could establish that the multi-parent diagonal crossover (6 parents variant tested in isolation) is better than the traditional 2-parent 1-point crossover on each test function. This is a new result in the sense that earlier papers considered performance of multi-parent crossovers on numerical optimization problems [6] and NK-landscapes [8]. The present study was performed on such bit-problems that have been carefully designed for studying and comparing crossovers in earlier studies, [9, 16, 17], and thus constitutes relevant new evidence on the viability of multi-parent recombination. It is interesting to note that the differences (in MBF) between the 6-parent and the 2-parent versions are by far the largest on the Royal Road

function. This function (or rather, family of functions) has been specifically designed to favor GAs using crossover above other search techniques, such as hill-climbers [16]. From our results it seems that on landscapes where crossover is supposed to be helpful, crossover using more parents is even more helpful.

In the light of the previous paragraph, it is clear that the answer to our second question, whether the adaptive or the standard GA performs better, depends on which crossover is used in the standard GA. The adaptive GA outperformed the standard GA using 1-point crossover, but showed comparable performance with the GA using diagonal crossover with 6 parents (Tables 6 and 7). These results suggest that it is sound and safe to use an adaptive GA with competing subpopulations/crossovers, instead of performing time consuming comparisons of several operators.

As for our third research objective we observed that the self-organizing mechanism was *not* able to modify the size of the sub-populations according to the performance of the crossover operators used in them when the generational gaps were kept equal (test series B and C). This conclusion is close to that of Spears' in [20]. In his work Spears used two crossovers and a self-adaptive mechanism to choose between them by encoding the applicable crossover in the chromosomes. The results were inconclusive and Spears conjectured that the GA wouldn't choose to use one of the crossovers, because the availability of two crossovers was better than just having one. This conjecture is in agreement with our results regarding the second research objective discussed above.

Further research is directed to other rewarding mechanisms of good crossovers in the competing subpopulations framework. Instead of enlargening successful subpopulations by adding individuals bred by other crossovers, increase in size can be done by giving them more space which they fill themselves. Research on such adaptive selection mechanisms is in progress and will be reported later.

References

[1] R.K. Belew and L.B. Booker, editors. *Proceedings of the 4th International Conference on Genetic Algorithms.* Morgan Kaufmann, 1991.

[2] J.P. Cohoon, W.N. Martin, and D.S. Richards. A multi-population genetic algorithm for solving the k-partition problem on hyper-cubes. In Belew and Booker [1], pages 244–248.

[3] Y. Davidor, H.-P. Schwefel, and R. Männer, editors. *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, number 866 in Lecture Notes in Computer Science. Springer-Verlag, 1994.

[4] L. Davis. Adapting operator probabilities in genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, 1989.

[5] K.A. De Jong and W.M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5:1–26, 1992.

[6] A.E. Eiben, C.H.M. van Kemenade, and J.N. Kok. Orgy in the computer: Multi-parent reproduction in genetic algorithms. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Advances in Artificial Life. Third International Conference on Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 934–945. Springer, Berlin, 1995.

[7] A.E. Eiben, P-E. Raué, and Zs. Ruttkay. Genetic algorithms with multi-parent recombination. In Davidor et al. [3], pages 78–87.

[8] A.E. Eiben and C.A. Schippers. Multi-parent's niche: n-ary crossovers on NK-landscapes. In Voigt et al. [23], pages 319–328.

[9] L.J. Eshelman and J.D. Schaffer. Crossover's niche. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 9–14. Morgan Kaufmann, 1993.

[10] D.B. Fogel and J.W. Atmar. Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems. *Biological Cybernetics*, 63:111–114, 1990.

[11] D.B. Fogel and L.C. Stayton. On the effectiveness of crossover in simulated evolutionary optimization. *Biosystems*, 32:171–182, 1994.

[12] R. Hinterding, Z. Michalewicz, and A.E. Eiben. Adaptation in evolutionary computation: A survey. In *Proceedings of the 4th IEEE Conference on Evolutionary Computation*, pages 65–69. IEEE, IEEE Press, 1997.

[13] IEEE. *Proceedings of the 3rd IEEE Conference on Evolutionary Computation*. IEEE Press, 1996.

[14] B. A. Julstrom. What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In S. Forrest, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 81–87. Morgan Kaufmann, 1995.

[15] J. Lis. Parallel genetic algorithm with dynamic control parameter. In *Proceedings of the 3rd IEEE Conference on Evolutionary Computation* [13], pages 324–329.

[16] M. Mitchell, S. Forrest, and J.H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In F.J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life*, pages 245–254. The MIT Press, 1994.

[17] J.D. Schaffer and L.J. Eshelman. On crossover as an evolutionary viable strategy. In Belew and Booker [1], pages 61–68.

[18] D. Schlierkamp-Voosen and H. Mühlenbein. Strategy adaptation by competing subpopulations. In Davidor et al. [3], pages 199–208.

[19] D. Schlierkamp-Voosen and H. Mühlenbein. Adaptation of population sizes by competing subpopulations. In *Proceedings of the 3rd IEEE Conference on Evolutionary Computation* [13], pages 330–335.

[20] W.M. Spears. Adapting crossover in evolutionary algorithms. In J.R. McDonnell, R.G. Reynolds, and D.B. Fogel, editors, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 367–384. MIT Press, 1995.

[21] B.A. Thijssen. *Adaptive Genetic Algorithms with Multiple Subpopulations and Multiple Parents.* Masters Thesis, Leiden University, 1997. Obtainable via `http://www.wi.leidenuniv.nl/MScThesis/bthijsse.html`.

[22] A. Tuson and P. Ross. Cost based operator rate adaptation: An investigation. In Voigt et al. [23], pages 461–469.

[23] H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors. *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, number 1141 in Lecture Notes in Computer Science. Springer, Berlin, 1996.

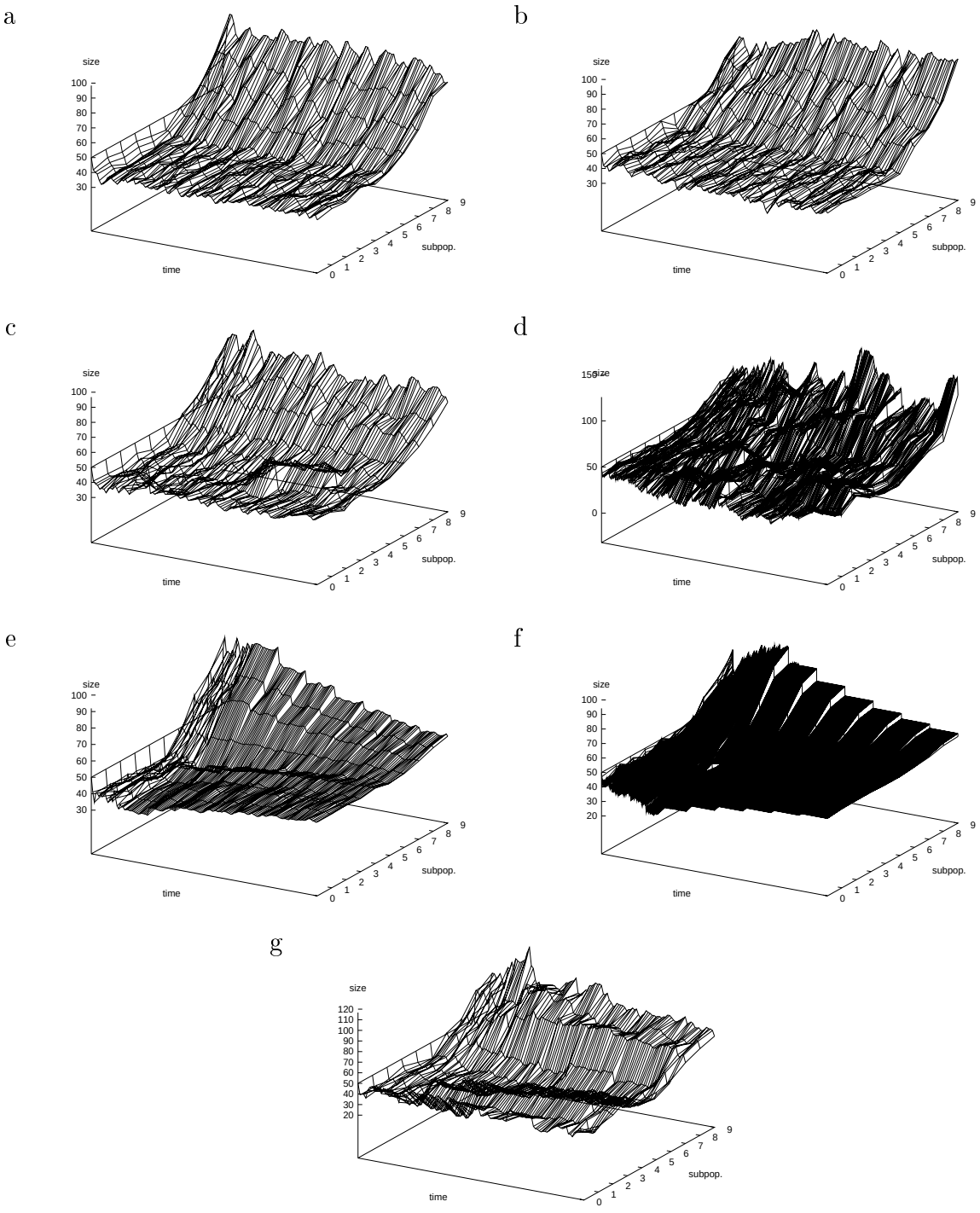

Figure 4: Test series A. Results for: a) the Onemax function, b) the Twin Peaks function, c) the Plateau function, d) the Plateau-d function, e) the Trap function, f) the Trap-d function, and g) the Royal Road function

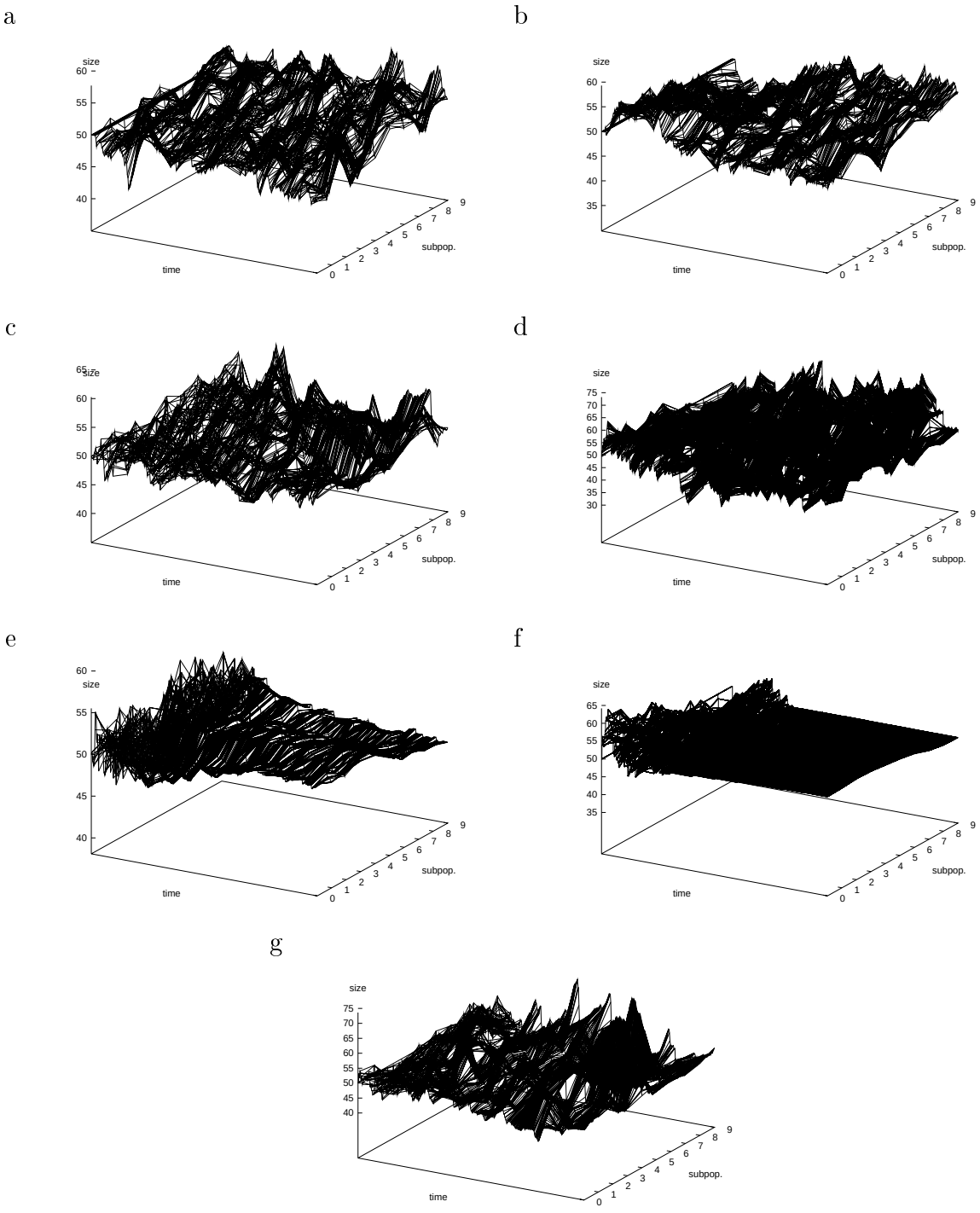

Figure 5: Test series B. Results for: a) the Onemax function, b) the Twin Peaks function, c) the Plateau function, d) the Plateau-d function, e) the Trap function, f) the Trap-d function, and g) the Royal Road function

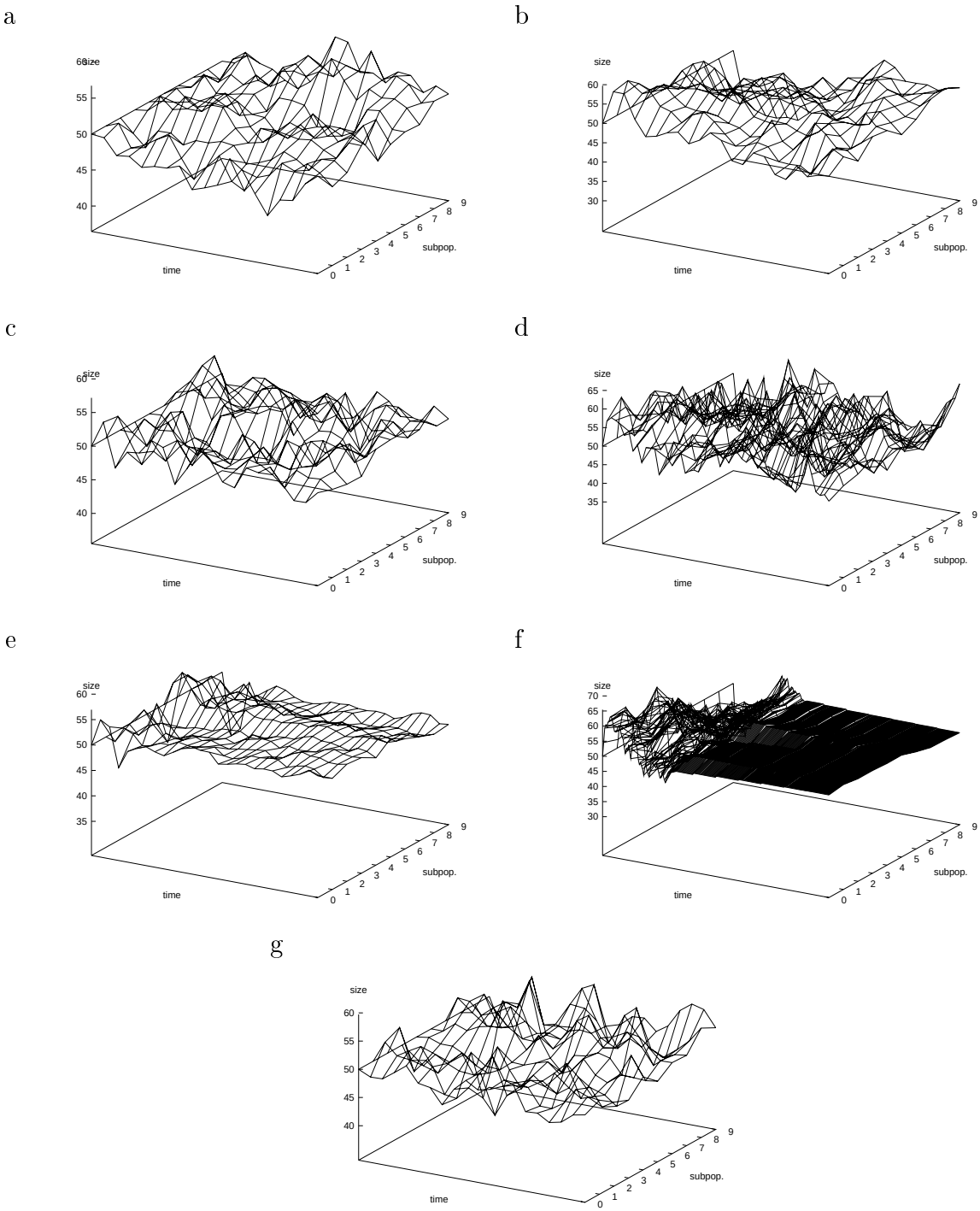

Figure 6: Test series C. Results for: a) the Onemax function, b) the Twin Peaks function, c) the Plateau function, d) the Plateau-d function, e) the Trap function, f) the Trap-d function, and g) the Royal Road function