Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Raising GA performance by simultaneous tuning of selective pressure and recombination disruptiveness

C.H.M. van Kemenade, J.N. Kok and A.E. Eiben

Computer Science/Department of Software Technology

CS-R9558 1995

Report CS-R9558
ISSN 0169-118X

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

Raising GA Performance by Simultaneous Tuning of Selective Pressure and Recombination Disruptiveness

C.H.M. van Kemenade

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

J.N. Kok and A.E. Eiben

Leiden University
Department of Computer Science
P.O. Box 9512, 2300 RA Leiden, The Netherlands
{*joost,gusz*}*@wi.leidenuniv.nl*

Abstract

In many Genetic Algorithms applications the objective is to find a (near-)optimal solution using a limited amount of computation. Given these requirements it is difficult to find a good balance between exploration and exploitation. Usually such a balance is found by tuning the various parameters (like the selective pressure, population size, the mutation- and crossover rate) of the Genetic Algorithm. As an alternative we propose simultaneous tuning of the selective pressure and the disruptiveness of the recombination operators. Our experiments show that the combination of a proper selective pressure and a highly disruptive recombination operator yields superior performance. The reduction mechanism used in a Steady-State GA has a strong influence on the optimal crossover disruptiveness. Using the worst fitness deletion strategy the building blocks present in the current best individuals are always preserved. This releases the crossover operator from the burden to maintain good building blocks and allows us to tune crossover disruptiveness to improve the search for better individuals.

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 International Conference on Evolutionary Computation, Perth 1995

1. Introduction

Genetic Algorithms have been applied to a variety of problems. In many practical applications of GA's the main objective is to find reasonable solutions within a reasonable amount of computation, where the exact meaning of reasonable differs per application. One interpretation of this statement is that one wants to find a (near-)*optimal* solution with a *high probability* using *limited computational efforts* (few function evaluations). Hence we have to find the proper balance between exploration and exploitation and we have to prevent premature convergence.

A typical application involves choosing a specific type of Genetic Algorithm, defining the operators, and tuning the parameters of the GA. These parameters are for example the probability of applying mutation, the probability of applying recombination, the size of the population, and the selective pressure. In GA's using linear ranking and tournament selection the selective pressure can be tuned by means of the ranking bias for linear ranking and the tournament size for tournament selection. Linear ranking and tournament selection fit nicely together in the sense that the maximal selective pressure of linear ranking corresponds to the minimal selective pressure of tournament selection, when populations are not too small.

We suggest that disruptiveness of the recombination operators is added to the list of important GA parameters. Disruptiveness of the recombination operator can have a strong influence on the search process, and we advice a *high* disruptiveness combined with an appropriate selective pressure in order to obtain a rapid GA.

One way of tuning the disruptiveness of recombination operators is to control the number of crossover points. De Jong et al. studied the influence of multi-point crossover [1] and Syswerda defined and tested the uniform crossover [5], which can be tuned using a biased coin. The generalized n-ary crossover operators [2, 3]

adjusts the number of parents in order to tune disruptiveness.

In this paper we focus on Steady-State GA's [8, 6] using a bit representation. This GA will be applied to function optimization problems, as these problems are often used as a test suite in GA related papers. However, we think that most statements in this paper are of a more general nature: they can be applied to other kinds of problems and other representations too. We will consider linear ranking and tournament selection as a selection mechanism, and we will combine these with random deletion and worst fitness deletion as a reduction mechanism. For the problems we consider it turns out that simultaneous tuning of selective pressure and recombination disruptiveness can result in a GA that outperforms GA's using the standard set of tunable parameters.

The paper is organized as follows. In section 2 we discuss different ways to speed up a GA. In section 3 we present an analysis of bit-variances and introduce a dynamic notion of disruptiveness. Then section 4 contains the experimental setup, and section 5 gives the results of the experiments. Finally, the conclusions are given in section 6.

2. GA CONVERGENCE VELOCITY

In many applications of GA's finding global optimal solutions requires far too much computation. In such cases it is necessary to find parameter settings that increase the convergence velocity of the GA, without resulting in too early premature convergence. An often applied method is minimizing the population size in order to obtain more rapid progress. Squeezing the population size too much can easily lead to stochastic sampling errors, as a smaller population carries less information. This results in premature convergence, leading to deterioration of the obtained solutions.

Another method to speed up a GA is increasing the selective pressure. As a result the GA is allocating more reproductive trials to fittest individuals in the population. It often is not advisable to make tournament sizes too large, as this might enlarge the probability of premature convergence. In order to understand the influence of the selective pressure let us have a look on the effect of the tournament size when applying tournament selection. Let α be the rank of a certain individual in the population consisting of n individuals and let b be the tournament size. We define $P_{\leq\alpha}(n,b)$ to be the probability that a selected individual has rank smaller than or equal to α:

$$P_{\leq\alpha}(n,b) = \left(\frac{\alpha}{n}\right)^b.$$

Hence the probability P_α (the element of rank α is selected) equals

$$P_\alpha(n,b) = \left(\frac{\alpha}{n}\right)^b - \left(\frac{\alpha-1}{n}\right)^b.$$

Figure 1 shows P_α as a function of the rank α (a higher rank corresponds to a fitter individual) and a pool size 200. The four curves shown correspond respectively to tournament sizes 2, 3, 4, and 5. As the tournament size increases the distribution concentrates near the better individuals and larger parts of the population are "shielded" from the evolution process. That is, when enlarging the tournament size the GA is focusing on a smaller part of the population. This results in a reduction of the effective population size, thereby again enlarging the probability of premature convergence.

To see the relation between linear ranking and tournament selection (at least for sufficiently large populations), we can use the following approximation

$$P_\alpha(n,b) \approx \frac{\partial}{\partial\alpha}\left(\frac{\alpha}{n}\right)^b = \left(\frac{1}{n}\right)^b b\alpha^{b-1}$$

If we take a tournament size $b = 2$ then we get the same distribution as for linear ranking with the ranking bias of 2, which is the maximal possible bias for linear ranking.

A less traditional method to speed up a GA is to increase the disruptiveness of the operators. A more disruptive operator can result in a better exploration of the search space. A disadvantage of a highly disruptive operator is that it easily leads to the loss of already discovered building blocks. An appropriate selection scheme can help in avoiding this kind of losses. The often used combination of fitness proportional selection and one-point crossover operator relies on making many copies of discovered building blocks in order to prevent the loss of already discovered building blocks. As evaluating (nearly-)identical individuals does not offer much new information, such a setting is inappropriate when rapid convergence is needed.

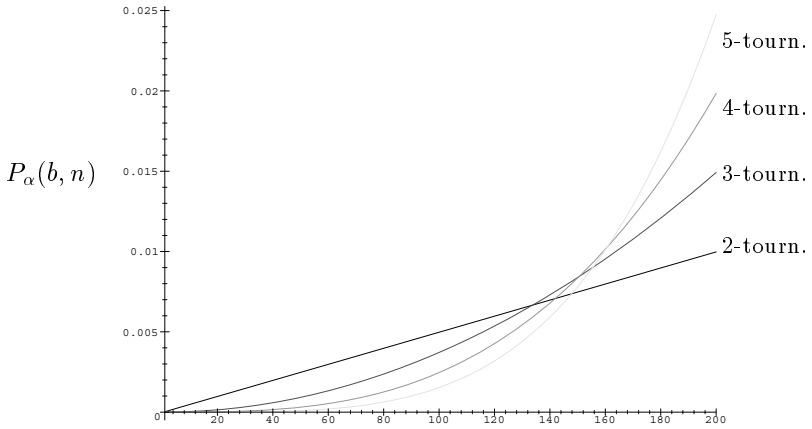

Figure 1: $P_\alpha(b,n)$ for $n = 200$ and tournament size = 2,3,4,5.

In the next section we investigate the role of disruptiveness in recombination operators. We do this by tracing vectors of bit variances and propose a new, dynamic definition of the probability of schema disruption.

3. Variances of Bits

Evolution is often visualized at phenotypic level, where one observes the development of average and current best fitness as a function of the number of function evaluations. In order to gain a better understanding of the behavior of recombination operators we traced the evolution process at genotypic level.

Given a population of bitstrings, one can calculate the expected value $E[a]$ and the variance $V[a]$ of each bit a over the population. Note that $V[a] = E[a] - E[a]^2$, since a bit can only take the value 0 or 1. For a random initial population of a reasonable size $E[a]$ will be 0.5 for each bit a. This implies that $V[a]$ is close to 0.25. In a population that has converged completely to one specific bitstring, the variances of all bits will be 0. Given a population of bitstrings we can assign a bit-variance vector by taking the variances of the successive bits. Bit convergence can than be visualized by tracing the evolution of this vector. During the evolution the variance of bits will decrease. However, this will not happen at the same rate for all bits. Some bits are more significant than other bits, in the sense that the specific value of some bits have much more influence on fitness than the value of other bits. The variance in the values of the most significant bits tends to decrease rapidly. When the variance in the value of a bit gets small, the probability that a less frequent value appears in a child gets small, assuming the mutation rate is not too high. The influence a certain bit a has on the fitness of the individual becomes more important as the values for bits that are more significant than bit a have converged. This way the significance of bits is reflected in a structure on the bit-variance landscape, obtained when tracing the evolution of the bit-variance vector.

As an example Figure 2 shows a sequence of bit-variance vectors for a GA trying to optimize a 2-dimensional Griewangk function. (The Griewangk function will be discussed in the next section.) The bitstring consists of a concatenation of two 15-bit fixed point integers, using binary-reflected Gray coding, where the left-most bit is the sign-bit, followed by the most significant bit. The back of this graph corresponds to the random initial population, where alle bit-variances are close to their maximal value.

When studying operators, their disruptiveness is an important aspect [4, 1]. In order to describe operator disruptiveness, one is often calculating the probability of schema disruption $P_d = \mathbb{P}[H_c \; and \; H_p]$, where H_c is the event that schemata H is not in the child and H_p is the event that schemata H appears in exactly one of the parents. Although many interesting observations regarding operator behavior can be obtained using this definition [1], we are more interested interaction between the recombination operator and a finite population, and therefore in the probability that a discovered schemata will survive in a population. So we are interested in $P_d = \mathbb{P}[H_c|H_q]$, where H_q is the event that a schemata is present in at least one of the

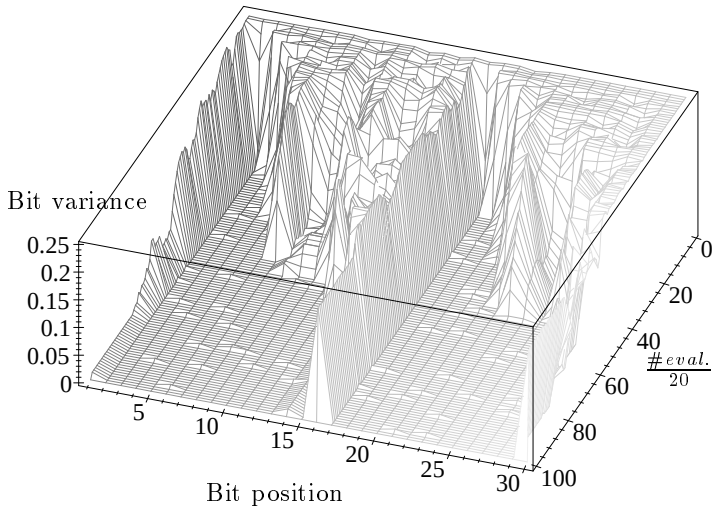

Figure 2: Evolving bit-variance vectors for a 30-bit two-dimensional Griewangk function.

parents. This can be rewritten as, $\mathbb{P}[H_c \text{ and } H_q]/\mathbb{P}[H_q]$. Here the numerator is a constant that differs per operator and the denominator changes due to the evolution of the population. When a GA is doing a good job and converges to the global optimum the quantity $\mathbb{P}[H_q]$ will either go to 1, if schemata H is present in the global solution, of $\mathbb{P}[H_q]$ will go to 0 if it is not. This means that the probability of disruption decreases for good schemata and increases for bad schemata, as the GA converges. This is exactly the behavior we need to find the solution.

By applying this more dynamic definition of the disruptiveness, that includes a notion of a changing population, we can gain new insights. For example, we can obtain intuition why disruptive crossover operators might work. Using the first (static) definition of P_d, it seems hard to imagine that a disruptive operator such as the uniform crossover can outperform a one-point crossover: bits that are close to one another are likely to belong to the same integer, and therefore are likely to be closely related, and uniform crossover seems to destroy these connections, and is unlikely to leave schemata of high order in intact. Using the second (dynamic) definition of P_d we get a different picture. Bit-variance vectors give a rough view of evolution at genotypic level, we can use such vectors to understand the influence of the changing population. If the variance in a certain bit is decreasing the probability of obtaining a value 0 or 1, when applying a recombination operator, is not equal anymore, so the preferred value of a bit appears. As a result good schemata seem to multiply themselves more easily than schemata resulting in a bad performance. Given the second definition of P_d, we see that a raising value for $\mathbb{P}[H_q]$, which corresponds to a schemata of relatively high fitness, will result in the recombination operator getting less disruptive for schemata H.

4. Set-up of the Experiments

We consider Steady-State GA's [8, 6]. During a single cycle of such a GA only a small fraction of the total population is replaced. In our experiments one child is produced, and replaces an individual of the population. A single cycle consist of selection, production and reduction. According to Syswerda [6] a Steady-State algorithm will behave almost identically to a Generational GA when random deletion is used as a reduction schedule. In many applications a worst fitness deletion is applied which causes a selective pressure, since the average fitness is likely to rise due to replacement of the worst performing individual by the new offspring, and good individuals have a relatively long lifetime.

We use tournament selection, as this mechanism is used in many applications because of its tunable selective pressure, that can vary over a large range. In order to change the selective pressure tournaments of different sizes ranging from 1 to 6 are used. A tournament size of 1 corresponds to uniform selection which

alone induces no selective pressure at all. A tournament size of 1 only makes sense when it is combined with a reduction mechanism that does induce selective pressure.

An individual in the population consists of the concatenation of fixed-point integers using a binary-reflected Gray coding. The population consists of 200 individuals. A mutation rate of $1/k$ is applied, where k is the length of the bitstrings. The number of function evaluations is limited to 20000. Recombination is applied always in order to prevent multiple copies from the same individual being created. If the Hamming distance between two parents is d then the standard recombination creates offspring that will be at a Hamming distance $d/2$ from the parents, on average. This is a much larger Hamming distance than can be expected from a mutation operator, as long as the parents are not almost identical.

Several recombination operators with tunable disruptiveness are available. We give three examples:

1. The n-point crossover. By increasing the number of crossover points this operator gets more disruptive [1].
2. The generalized n-ary recombination operators, with adjustable arity, as introduced in [2, 3].
3. The uniform crossover of Syswerda [5] can be tuned by using a biased coin ($P_0 \neq 0.5$) [1].

In the experiments we are going to apply the diagonal crossover (a generalized n-ary version of the one-point crossover) and n-point crossover. The disruptiveness of this operator can be tuned easily by adjusting its number of parents. Diagonal crossover with arity p selects $(p-1)$ distinct crossover points resulting in p chromosome segments in each of the p parents and composes p offspring by taking the pieces from the parents "along the diagonals". For $p = 2$ diagonal crossover coincides with the traditional 1-point crossover. We use a modified version of diagonal crossover produces just one offspring, in order to reduce the sampling error.

In the experiments we used two standard function optimization problems, which are known to be difficult: the 10-dimensional Schwefel function and the 10-dimensional Griewangk function. The Schwefel function is defined as:

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

where $-500 \leq x_i \leq 500$. The global minimum of zero is obtained for $\vec{x} = (420.9687, 420.9687, \cdots)$. In the Schwefel function the best and second best optimum are far apart.

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 obtained for $\vec{x} = (0, 0, 0, \cdots)$. One of the characteristics of the Griewangk function is that the different dimensions can not be optimized independently of one another.

All experiments are repeated 25 times.

5. Results of the Experiments

Figure 3 and 4 show the rate of success when optimizing the 10-dimensional Schwefel function. A run is assumed to be successful if the fitness of the overall best individual found is less than 10^{-5}. The two horizontal axes show the tournament size and the number of parents. A higher tournament size corresponds to a higher selective pressure and a higher number of parents corresponds to a higher disruptiveness. Figure 3 shows the results when using random deletion, and Figure 4 shows the results when applying worst fitness deletion.

Table 1 contains a brief summary of results obtained when optimizing the Schwefel function using the modified diagonal crossover. Table 2 contains the same summary for a GA using the n-point crossover. When using a random deletion schedule, which most closely corresponds to a generational GA, a large tournament size is needed to in order to find the global optimum. This high tournament size is necessary as all individuals have the same probability of being deleted ($1/n$, where n is the size of the population). In order to prevent the loss of discovered good building blocks probably present in the best performing individuals, a high bias towards these individuals is needed. This high tournament size results in many

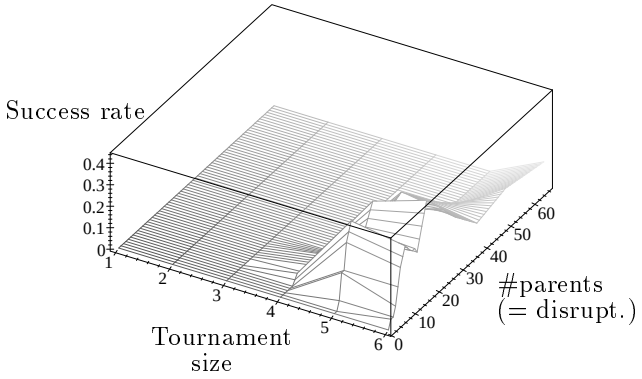

Figure 3: Schwefel function optimized by a Steady-State GA with tournament selection and random deletion using the modified diagonal crossover. A success rate $\geq 35\%$ was obtained when using a 6-tournament and 17-22 parents during crossover.

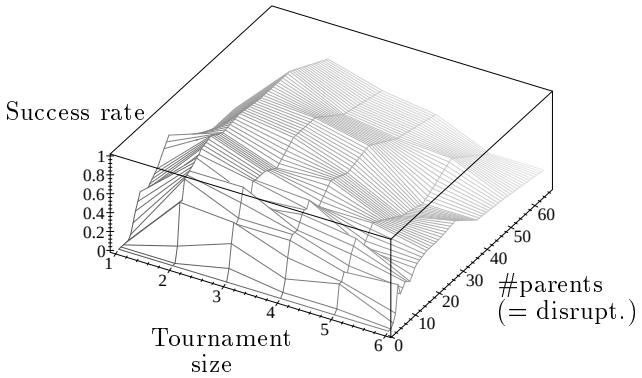

Figure 4: Schwefel function optimized by a Steady-State GA with tournament selection and worst fitness deletion using the modified diagonal crossover. A success rate $\geq 90\%$ was obtained when using a 2-tournament and 13-50 parents during crossover.

(partial) copies of the best few individuals being created. When a worst deletion schedule is applied, the lifetime of an individual is strongly related to its relative fitness within the current population. In that case relatively fit individuals never get lost, and there is no reason to assign multiple copies to such individuals. In fact, assigning multiple copies is a waste of computation, and might lead to an allocation of too many reproductive trials to the multiplied individuals [9], and therefore lead to premature convergence. For this reason the worst fitness deletion mechanism is best combined with a low tournament size.

The same set of experiments has been applied to optimize the Griewangk function. Here a run is assumed to be successful if the fitness of the overall best individual found is below 10^{-2}. Table 3 shows the results

	Random Del.		Worst Del.	
Selection	#par.	success r.	#par.	success r.
Unif. sel.	—	0%	17–29	$\geq$45%
2-tourn.	—	0%	13–50	$\geq$90%
6-tourn.	17–22	$\geq$35%	$\pm$22	$\geq$70%

Table 1: Modified diagonal crossover: best amount of disruptiveness for different combinations of the selection and deletion schedules on Schwefel function (#par. = number of parents).

Selection	Random Del. #Xp.	Random Del. success r.	Worst Del. #Xp.	Worst Del. success r.
Unif. sel.	—	0%	—	0%
2-tourn.	—	0%	5–28	≥36%
6-tourn.	3,7,12...	≥12%	±9	24%

Table 2: N-point crossover: best amount of disruptiveness for different combinations of the selection and deletion schedules on Schwefel function (#Xp. = number of crossover points).

Selection	Random Del. #par.	Random Del. success r.	Worst Del. #par.	Worst Del. success r.
Unif. sel.	—	0%	10–50	≥20%
2-tourn.	—	0%	50–117	≥15%
6-tourn.	6,8,29...	4%	2,6,10...	8%

Table 3: Modified diagonal crossover: best amount of disruptiveness for different combinations of the selection and deletion schedules on Griewangk function (#par. = number of parents).

when applying the modified diagonal crossover and Table 4 shows the results when applying the n-point crossover. Again we see that a low selective pressure combined with reasonably high disruptiveness of the recombination operator seems the appropriate setting. The best results are obtained when applying diagonal crossover using a number of parents somewhere between 13 and 50, but this time a tournament size of 1, corresponding to uniform selection, performs even better than a tournament size of 2. When applying an n-point crossover we see a preference for a high selective pressure. Unfortunately we can not draw more conclusions as the number of times a GA applying n-point crossover converges is too low.

In order to get a better comparison to other GA's we also performed a set of tests using a standard Generational GA. This GA uses 2-tournament selection and a 1-point or 9-point crossover with probability of $P_c = 0.7$. All other parameters are equal to those used in the Steady-State GA. Figure 5 shows the average best individual as a function of the number of function evaluations. The curves are again obtained by taking averages over 25 independent runs. The Generational GA converges much slower than the Steady-State GA. After approximately $2 \cdot 10^4$ function evaluations the best individual starts even to deteriorate. This is probably the result of cross-competition between the well performing building blocks of short defining length [7]. When the diversity in these building blocks decreases the probability the current best individual is present in subsequent generations starts to decrease too. When applying a more disruptive crossover, such as the 9-point crossover, this deterioration appears even earlier. When optimizing the Griewangk function we also see slow initial convergence and deterioration of the population when convergence stagnates. In order to prevent this deterioration we also did some tests using the Generational GA with a population size of 1000 instead of 200. Using such a large population size, the success rate is 32%, but the optimum is never obtained in less than $2 \cdot 10^5$ function evaluations.

Selection	Random Del. #Xp.	Random Del. success r.	Worst Del. #Xp.	Worst Del. success r.
Unif. sel.	—	0%	—	0%
2-tourn.	—	0%	21,49	4%
6-tourn.	2,3	4%	1,16,49,64	4%

Table 4: N-point crossover: best amount of disruptiveness for different combinations of the selection and deletion schedules on Griewangk function (#Xp. = number of crossover points).

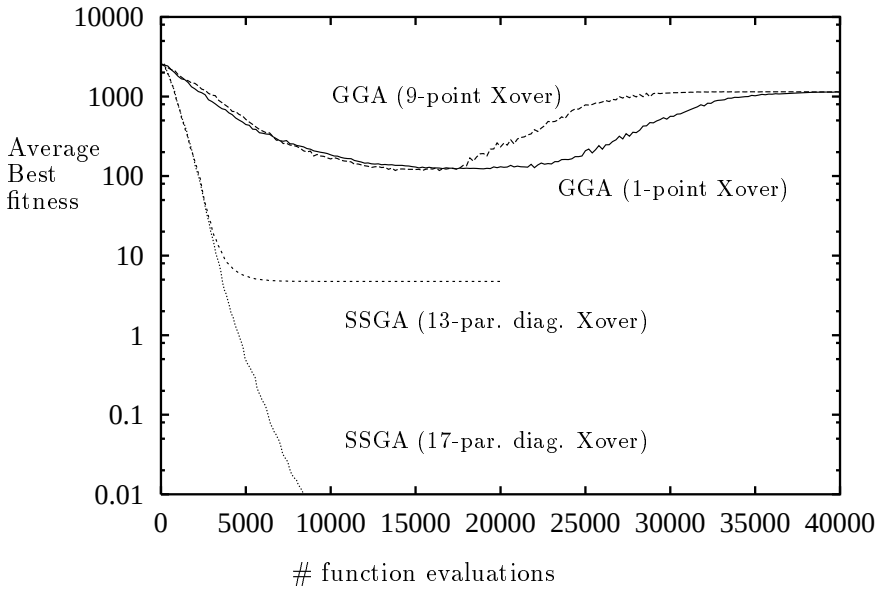

Figure 5: Comparison between a Generational GA (GGA) 1-point or 9-point crossover ($P_c = 0.7$) and a Steady-State GA (SSGA) using 13-parent or 17-parent modified diagonal crossover ($P_c = 1$) on the Schwefel function. Both GA's use a population size of 200.

6. Conclusions

The disruptiveness of a recombination operator is usually expressed as a static value that only depends on the schema and the operator in consideration. A more sophisticated picture of schema survival can be obtained by using a dynamic measure of disruptiveness as we introduced in section 3. This definition includes the interaction between an operator and the population it is applied to. Tracing evolution at a genotypic level by means of bit-variance vectors we can get a good picture of the role of disruptiveness in recombination operators. Furthermore, the differences between disruptiveness in a recombination and in a mutation operator become clear by this view. In particular, recombination disruptiveness for a schema H depends upon the probability that schema H is present in at least one of the parents. For this reason a (disruptive) recombination can help in increasing population diversity and preventing multiple copies from arising without harming the search process. However, a very disruptive (bit flipping) mutation operator will change the GA in something like a randomized hillclimber.

We observed that there is a correlation between the optimal values of selective pressure and recombination disruptiveness. GA's that enforce a low selective pressure often require a recombination operator which has a low disruptiveness in order to preserve discovered good schemata. Higher selective pressures allow the application of more disruptive recombination operators. As one increases selective pressure it even becomes necessary to use a more disruptive recombination operator, in order to prevent premature convergence. For this purpose we need disruptiveness in the recombination operator, as the recombination is better able to maintain discovered schemata than a mutation operator.

In practical applications one often needs a problem solver that finds reasonable solutions using a limited amount of computation. When applying a GA with this objective in mind, a Steady-State GA using worst fitness deletion and a disruptive recombination operator is our preferred combination. Furthermore, instead of tuning the probability of crossover we set to $P_c = 1$ and tune the recombination disruptiveness.

References

1. 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.

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

3. 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.
4. 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.
5. G. Syswerda. Uniform crossover in genetic algorithms. In *Third International Conference on Genetic Algorithms*, pages 2–9, 1989.
6. G. Syswerda. A study of reproduction in generational and steady–state genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms - 1*, pages 94–101. Morgan Kaufmann, 1991.
7. D. Thierens and D.E. Goldberg. Mixing in genetic algorithms. In S. Forrest, editor, *Fifth International Conference on Genetic Algorithms*, pages 38–45. Morgan Kaufmann, 1993.
8. D. Whitley. The GENITOR algorithm and selective pressure. In *Third International Conference on Genetic Algorithms*, pages 116–121, 1989.
9. D. Whitley and T. Starkweather. GENITOR II: A distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:189–214, 1990.