Pergamon

Mathl. Comput. Modelling Vol. 21, No. 8, pp. 1–11, 1995

0895-7177/95 $9.50 + 0.00

0895-7177(95)00035-6

Optimal Mutation Probability for Genetic Algorithms

R. N. GREENWELL
Hofstra University
Hempstead, NY 11550, U.S.A.

J. E. ANGUS
Department of Mathematics
The Claremont Graduate School
143 E. Tenth St., Claremont, CA 91711, U.S.A.

M. FINCK
Hofstra University
Hempstead, NY 11550, U.S.A.

(Received January 1995; accepted February 1995)

Abstract—We derive the value of the mutation probability which maximizes the probability that the genetic algorithm finds the optimum value of the objective function under simple assumptions. This value is compared with the optimum mutation probability derived in other studies. An empirical study shows that this value, when used with a larger scaling factor in linear scaling, improves the performance of the genetic algorithm. This feature is then added to a model developed by Hinton and Nowlan which allows certain bits to be guessed in an effort to increase the probability of finding the optimum solution.

Keywords—Optimization, Natural selection, Genetic search, Fitness scaling, Global extrema.

1. INTRODUCTION

In genetic algorithms, mutation is often regarded as a background operator whose only importance is to prevent the algorithm from prematurely converging to a suboptimal solution. Numerous studies have investigated the optimal setting for the mutation probability p_m [1–7]. In this paper, we take a different approach to the problem, namely, choosing p_m to maximize the probability that the genetic algorithm finds the optimum solution. In other words, as selection and crossover introduce better solutions, mutation is used in hopes of nudging good solutions closer to the best solution. This approach bears some resemblance to that of Hinton and Nowlan [8] and Belew [9], who allowed certain undecided bits to be guessed in an attempt to find the optimum solution. We will investigate the addition of a mutation probability to their model.

The original motivation for genetic algorithms was adaptation of a population, and so preservation of good schemata in the gene pool argued against a large mutation rate. This tradition does not preclude considering a genetic algorithm in the narrow sense of a function optimizer, without regard to what happens to the population. Genetic algorithms have proven useful in this narrow sense, so in this paper we restrict our attention to the goal of function optimization. We also consider only unimodal functions; if there are multiple points where the optimum occurs, we seek only one such point.

2. THE OPTIMAL MUTATION PROBABILITY

Without loss of generality, let us assume the optimal solution to a genetic algorithm problem is a string of L 1's (assume $L > 1$). Denote this solution by X. Suppose we have a population of size N, in which the probability that an arbitrary bit is 1 is denoted by p. For simplicity, we assume that the value of a bit is independent of the value of any other bit. Although this is too simplistic for many problems, it gives us a starting point. Denote by E_X the event that at least one population member is X after mutation (with E_X^c its complimentary event), and by F_X the event that none of the population members were X before mutation. Then, whenever $p \in [0, 1)$,

$$\begin{aligned} P(E_X^c \mid F_X) &= [P(\text{a population member is not X after mutation} \mid F_X)]^N \\ &= [1 - P(\text{a population member is X after mutation} \mid F_X)]^N \\ &= [1 - P(\text{all bits are 1 after mutation} \mid \text{at least one bit was 0 before})]^N \\ &= \left(1 - \frac{\sum_{k=1}^{L} \binom{L}{k}(1-p)^k p^{L-k} p_m^k (1-p_m)^{L-k}}{1-p^L}\right)^N \\ &= \left(1 - \frac{(p_m(1-2p)+p)^L - (p(1-p_m))^L}{1-p^L}\right)^N, \end{aligned}$$

and hence the probability that at least one population member is X after mutation given that none is X before mutation is

$$P(E_X \mid F_X) = 1 - \left(1 - \frac{(p_m(1-2p)+p)^L - (p(1-p_m))^L}{1-p^L}\right)^N. \tag{1}$$

For fixed p, L, and N, denote this last expression by $f(p_m)$. Consider the function (for fixed p and L) $h(p_m) = (p_m(1-2p)+p)^L - (p(1-p_m))^L$. There is only one solution to the equation $h'(p_m) = 0$ when $p \in [1/2, 1]$, and none otherwise. Denoting this solution by p_m^*, we find that

$$p_m^* = \frac{1 - p\alpha}{(1-2p)\alpha + 1}, \tag{2}$$

where

$$\alpha = \left(\frac{2p-1}{p^L}\right)^{1/(L-1)}. \tag{3}$$

It is straightforward to verify that $0 \le p_m^* \le 1$ for all $p \in [1/2, 1]$. Now $h'(0) > 0$ and $h(0) = 0$ for all $p \in [0, 1]$. Also, $h'(1) < 0$ for all $p \in (1/2, 1)$, and $h(1) = (1-p)^L > 0$ for all $p \in [0, 1)$. Hence, as long as $p \in (1/2, 1)$, $h(p_m) > h(1)$ for some $p_m \in (1-\delta, 1)$ for some $\delta > 0$. Hence, for any $p \in (1/2, 1)$, p_m^* must yield a global maximum of f on $p_m \in [0, 1)$, since it is the only critical point there. When $p = 1/2$, $p_m = 1$ maximizes f, so (2) maximizes f for all $p \in [1/2, 1)$.

Notice that (2) and (3) can be expressed in the alternate form

$$p_m^* = \frac{1-u}{1-u^L}, \tag{4}$$

where

$$u = \left(\frac{2p-1}{p}\right)^{1/(L-1)}. \tag{5}$$

As p increases from 1/2 to 1, u increases continuously from 0 to 1. Hence,

$$\lim_{p \to 1-} p_m^* = \lim_{u \to 1-} \frac{1-u}{1-u^L} = \frac{1}{L}.$$

Moreover, as $p \uparrow 1$, $p_m^* \downarrow 1/L$. To verify this, it suffices to verify that $(1-u)/(1-u^L) \downarrow 1/L$ as $u \uparrow 1$. But this follows because for $u \in (0,1)$, $0 < u^L + Lu^{L-1}(1-u) < \sum_{k=0}^{L} \binom{L}{k}(1-u)^k u^{L-k} = 1$, so that

$$\frac{d\left((1-u)/(1-u^L)\right)}{du} = \frac{\left(u^L + Lu^{L-1}(1-u) - 1\right)}{(1-u^L)^2} < 0.$$

In practice, one doesn't know p, making use of (2) and (3) problematic. One can estimate p using

$$\hat{p} = 1 - \frac{1}{N-1}\sum_{i=1}^{N}\frac{h_i}{L}, \tag{6}$$

where h_i is the Hamming distance between element i of the population and the best solution found so far.

The limiting value p_m^*, namely $1/L$, has been previously recommended by DeJong [2]. If population size is kept constant, this value is asymptotically smaller than the optimum found by Schaffer *et al.* of [7] $p_m = 1.829N^{-1.073}L^{-.4867}$. But if the population size is increased along with the chromosome length, our result is asymptotically larger. Schaffer *et al.*, however, were using the criterion of online performance, which is the average value of the objective function over all members of the population. This study, on the other hand, is concerned solely with finding the best solution at any given generation.

The goal of hoping to mutate one's way to the optimal solution may well be questioned, because for large L, $f(p_m^*)$ is very small. But $\lim_{N\to\infty} f(p_m^*) = 1$, which says that if the population is made large enough, the probability of hitting the solution can be made arbitrarily close to 1. Using L'Hôpital's Rule, it may be verified that

$$\lim_{p\to 1} f\left(\frac{1}{L}\right) = 1 - \left(1 - \frac{(L-1)^{L-1}}{L^L}\right)^N.$$

This tells us the probability of hitting the optimal solution as the population gets close to that solution. For example, when $L = N = 30$, this limit is 0.3137.

3. EMPIRICAL RESULTS

We have tested the results of the previous section running Goldberg's SGA [10] for 60 generations with the values $L = N = 30$. The first objective function used was the number of 1 bits, raised to the fifteenth power, and then divided by 2^L, with linear scaling used to determine fitness. It was chosen because of its single sharp peak around the optimum solution. For this fitness function, the genetic algorithm usually found the optimum solution before the 60^{th} generation. Therefore, our criterion for comparison is the number of generations required before the optimum solution was found.

In our initial experiments, we compared scaling multiples in the range of 1.2 to 2.0, which Goldberg recommends, and values several times larger. We found that larger scaling factors led to improved performance, and so we eventually tried scaling factors as large as 30, even though this is far outside the usually recommended values. Goldberg's SGA changed to an even smaller scaling factor when necessary to avoid negative fitness. In the current study, any negative fitness values were simply changed to 0. In addition to the fact that this strategy worked well, the following rationale may be given: if the mutation probability is large enough to prevent premature convergence, it is worthwhile to strongly reward good solutions and strongly penalize bad solutions. Mutation will keep shaking things up enough so that other parts of the solution space will be explored. This strategy bears some resemblance to the dynamic, extinctive selection scheme described by Bäck and Hoffmeister [11].

We tried six different values of p_m: 0.0091 recommended by Schaffer *et al.* [7], 0.011 recommended by Hesser and Männer [5], and the changing value of p_m^* given by (2), (3) and (6), plus

the unusually large values of 0.2 and 0.1 and the minimum value of 0.0. Uniform crossover was used with the crossover probability kept at 0.6. Each combination of parameters was tried 99 times.

The following ANOVA table shows that the scaling multiple and the mutation probability, as well as the interaction between them, were extremely significant.

	df	SS	MS	*F*	*P*-value
Scaling	29	694685	23955	386.37	0.0000
Mutation	5	7143257	1428651	23043	0.0000
Interaction	145	578930	3993	64.403	0.0000
Error	17640	1097567	62		
Total	17819	9514440			

Appendix 1 shows the average number of generations (over 99 trials) until the optimum solution was found. The mutation probabilities of 0.2, 0.1, and 0.0 gave poor results, while the other three rates gave much better results. The best results averaged over all scaling factors (14.193) were found using p_m^*. In particular, the combination of p_m^* with the scaling factor of 30 gave the best performance (9.343). One striking feature of the table is that the results continued to improve as the scaling factor increases, except with a mutation rate of 0.0. When using the traditional scaling factor 2 or less, the mutation rates of 0.011 and 0.0091 out perform the rate given here, but this reverses for scaling factors larger than 4.

One possible difficulty with a large scaling factor is premature convergence to a local optimum that is not the global optimum. To investigate this, we created a second fitness function by adding to the original function the number of 0 bits, raised to the ninth power, and then divided by 2^L and multiplied by 0.5. This function has local optimum of 0.5 when all bits are set to 0. The ninth power was chosen because it was sufficiently smaller than 15, the power used in the original function, that the algorithm often found the local optimum of 0.5 (when all bits are 0) rather than the global optimum of 1.0 (when all bits are 1). For this reason, rather than looking at the number of generations until the optimum was found, we instead looked at the fitness of the best individual found in 60 generations. As the following ANOVA shows, the mutation rate, scaling factor, and interaction are highly significant.

	df	SS	MS	*F*	*P*-value
Scaling	29	73.1587	2.5227	2.5097	0.0000
Mutation	5	348.1180	69.6236	7591.23	0.0000
Interaction	145	31.0458	0.2141	23.3438	0.0000
Error	17640	161.7867	0.00917158		
Total	17819	614.1093			

Appendix 2 shows the average fitness (over 99 trials) of the best individual found in 60 generations. The mutation probabilities of 0.2, 0.1, and 0.0 again gave poor results. There was little difference between the results for the other three mutation rates. The results do not show the striking improvement shown in Appendix 1 as the scaling factor increases.

We next asked whether these new values of the parameters would work for more interesting problems, where our original assumptions do not apply. We tried the function F6 defined by Schaffer *et al.* [7] as

$$F6(x,y) = 0.5 + \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{[1+0.0001(x^2+y^2)]^2}.$$

As before, we let $L = 30$; we split the 30 bits into two blocks of 15 bits each, with 1 bit representing the sign, 4 bits representing the bits to the left of the decimal, and the other 10 bits representing the bits to the right of the decimal. The results below show that the scaling factor was highly significant and the mutation index extremely so, while the interaction between them was not.

	df	SS	MS	*F*	*P*-value
Scaling	29	0.022575	0.000778	2.5097	0.0000
Mutation	5	2.027066	0.405413	1308	0.0000
Interaction	145	0.034344	0.000237	0.7645	0.9839
Error	17640	5.466846	0.000310		
Total	17819	7.550830			

Appendix 3 shows the average fitness (over 99 trials) of the best individual found in 60 generations. All mutation rates gave good results this time, but the best results were for 0.2 and 0.1, followed by p_m^*. In this case, even higher mutation rates may be useful. Although the scaling factor is statistically significant, its value does not seem to make much of a difference.

Davis [12] has argued that F6 and the other functions in Schaffer's test suite are not suitable for testing genetic algorithms because they favor what he terms "bit climbers." He suggests instead an alternative test suite in which the value of each variable is shifted down by 10%. The shifted version of F6 is referred to as SF6. We repeated our previous experiment using SF6 with uniform crossover. As the results below show, mutation was statistically significant, but scaling was not, nor was the interaction between the two.

	df	SS	MS	*F*	*P*-value
Scaling	29	0.002029	0.000070	0.44300	0.9958
Mutation	5	0.965771	0.193154	1220	0.0000
Interaction	145	0.015066	0.000104	0.65823	0.9995
Error	17640	2.782971	0.000158		
Total	17819	3.765838			

Appendix 4 shows the average fitness (over 99 trials) of the best individual found in 60 generations. As with F6, all mutation rates gave good results, with the best results for 0.1, followed by 0.2, followed by p_m^*. The scaling factor does not matter.

These results are inconsistent with those of Schaffer *et al.* [7]. The primary difference between our work and that of Schaffer *et al.* is that we looked for the best solution found rather than the online performance. It may be that too much attention has been given to online and offline performance. To someone with a problem requiring a solution, the only matter of interest is how good is the solution found. We note that Bramlette [1] and Laszewski [13] have found even larger mutation rates to be useful when seeking the global optimum.

4. THE MODEL OF HINTON, NOWLAN, AND BELEW

Hinton and Nowlan [8] and Belew [9] suggested adding "shoulders" to an otherwise isolated optimum solution by considering bits with values 0, 1, and ?. The idea is that the genetic algorithm will have a greater chance of finding the optimal solution if we allow a large number of guesses for certain bits (those denoted by ?'s). The model is therefore of interest in this study because it, too, has the goal of finding the optimal solution by chance. The model does not contain a mutation probability, so we propose to add this feature.

As in [9], let Q denote the number of bits in a member of the population which are to be guessed, let G be the number of guesses allowed for the string, and let r denote the probability

that a bit is 0. As before, let p_m denote the probability that a 1 mutates to a 0 or a 0 to a 1. The ? bits will not be allowed to mutate. Then as a function of the mutation probability p_m (and fixed G, Q, L), the probability that a member of the population is the optimum after mutation and guessing is given by

$$g(p_m) = \left(1 - \left(1 - 2^{-Q}\right)^G\right) \sum_{k=0}^{L-Q} \binom{L}{k} r^k (1-r)^{L-k} p_m^k (1-p_m)^{L-Q-k}. \tag{7}$$

Here k represents the number of 0's, which must mutate, and $L - Q - k$ represents the number of 1's, which must not mutate if we are to achieve the optimal solution. The factor in the front of the summation is the probability that the ? bits are guessed correctly at least once out of the G guesses, while the summation represents the probability that any 0's in a string change to 1 and the 1's stay as they are. When p_m is 0, this formula is identical to Belew's. If Q set to 0 and no ? bits are allowed, then $r = 1 - p$, and (7) is the same as the numerator of the fraction in the fourth line of the derivation of (1), except that in (1) we did not allow k to equal 0, since we were assuming that the population didn't already contain the optimal solution. On the other hand, when $Q > 0$, the population automatically does not contain the optimal solution because it does not contain any particular solution until the ? bits are guessed.

Let us seek the value of p_m that maximizes the probability in (7). Solving $g'(p_m) = 0$ is algebraically very complicated. Furthermore, in the special cases we investigated, the solution turned out to be a minimum. If this is always true, then the maximum must occur at either $p_m = 0$ or $p_m = 1$. Then it can be shown that when

$$r < \frac{1}{1 + \binom{L}{Q}^{1/(L-Q)}}, \tag{8}$$

the maximum occurs at $p_m = 0$. When r is greater than the value in (8), the maximum occurs at $p_m = 1$. In other words, if few of the bits are incorrect, it's better if none of them mutate. Otherwise it's better if all of them mutate (except for those which are to be guessed).

This result is quite different from the intermediate value of p_m^* given by (2) and (3). That is because we are now leaving open the possibility that all the determined bits are already 1, a case that was excluded previously. In Section 2, had we summed on k from 0 to L rather than from 1 to L, we would also find the optimum value of p_m to be 0 or 1. This implies that if there is a likelihood that the optimum already exists in the population, no mutations should be made when our goal is maximizing the probability of finding the optimum. On the other hand, if all members of the population are very far from the best solution, it might be best to change the values of all bits.

Making p_m equal to 0 or 1 is not a good strategy in the standard genetic algorithm. Clearly more work is needed to unify the ideas of this section with the ideas of Section 2. For example, (1) deals with an entire population, while (7) deals only with one member of the population. If (7) is expanded to an entire population, then Q must be a random variable if one adheres to Belew's formulation. In that case, deriving a result similar to (7) is complicated.

5. CONCLUSIONS

This study suggests that using a mutation rate different from what has been suggested by others increases the chance that the genetic algorithm will find the optimum solution, and improves the value of the best solution found even when the optimum solution is not found. In contrast, when this idea is added to the model of the genetic algorithm which allows certain bits to be guessed, we come to the conclusion that it is best to change all bits or none of the bits, neither of which is a good strategy in the standard genetic algorithm.

REFERENCES

1. M.F. Bramlette, Initialization, mutation and selection methods in genetic algorithms for function optimization, In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 100–107, Morgan Kaufmann, (1991).
2. K. DeJong, Analysis of the behavior of a class of genetic adaptive systems, Ph.D. dissertation, University of Michigan, (1975).
3. T.C. Fogarty, Varying the probabilities of mutation in the genetic algorithm, In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 104–109, Morgan Kaufmann, (1989).
4. J.J. Grefenstette, Optimization of control parameters for Genetic Algorithms, *IEEE Transactions on Systems, Man and Cybernetics* **SMC-16**, 122–128 (1986).
5. J. Hesser and R. Männer, Towards an optimal mutation probability, In *Proceedings of the International Workshop Parallel Problem Solving from Nature*, Springer-Verlag, (1990).
6. M. Nowack and P. Schuster, Error thresholds of replication in finite populations mutations frequencies and the onset of Muller's Ratchet, *Journal of Theoretical Biology* **137**, 375–395 (1989).
7. J.D. Schaffer, R.A. Caruna, L.J. Eshelman and R. Das, A study of control parameters affecting on line performance of genetic algorithms for function optimization, In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 51–60, Morgan Kaufmann, (1989).
8. G.E. Hinton and S.J. Nowlan, How learning can guide evolution, *Complex Systems* **1**, 495–502 (1987).
9. R.K. Belew, When both individuals and populations search: Adding simple learning to the Genetic Algorithm, In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 34–41, Morgan Kaufmann, (1989).
10. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, (1989).
11. T. Bäck and F. Hoffmeister, Extended selection mechanisms in genetic algorithms, In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 92–99, Morgan Kaufmann, (1991).
12. L. Davis, Bit-climbing, representational bias, and test suite design, In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 18–23, Morgan Kaufmann, (1991).
13. G. von Laszewski, Intelligent structural operators for the k-way graph partitioning problem, In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 45–52, Morgan Kaufmann, (1991).

APPENDIX 1
ONEMAX

Number of generations until optimum found.
Rows: fitness scaling factor.
Columns: mutation probability.

	0.2	0.1	p_m^*	0.011	0.0091	0.0	all
1	60.000	60.000	60.000	60.000	60.000	60.000	60.000
2	60.000	60.000	60.000	31.859	33.202	57.697	50.460
3	60.000	60.000	27.404	18.081	18.061	57.576	40.187
4	60.000	59.788	15.970	15.141	16.061	57.495	37.409
5	60.000	56.030	13.101	14.374	15.020	58.444	36.162
6	60.000	51.071	11.717	13.313	14.374	58.475	34.825
7	59.606	44.384	10.980	13.707	14.717	56.323	33.286
8	60.000	42.071	10.889	12.596	13.333	58.939	32.971
9	59.646	39.727	10.535	12.515	13.626	58.909	32.493
10	59.949	38.646	10.384	12.677	14.343	59.465	32.577
11	59.717	38.343	10.091	13.273	13.788	58.444	32.276
12	60.000	36.960	10.040	12.293	13.545	57.818	31.776
13	59.182	35.313	10.020	12.384	13.626	58.374	31.483
14	60.000	35.101	9.990	12.071	13.293	57.828	31.380
15	59.677	32.182	9.737	12.455	12.859	58.354	30.877
16	60.000	33.343	9.828	12.434	13.535	59.444	31.431
17	60.000	33.970	9.707	12.293	13.000	60.000	31.495
18	59.929	33.152	9.929	12.566	13.253	60.000	31.471
19	59.606	34.717	9.626	12.707	13.303	60.000	31.660
20	60.000	33.889	9.414	12.152	13.455	60.000	31.485
21	59.545	32.091	9.687	12.040	13.293	59.465	31.020
22	59.616	32.091	9.586	12.071	13.071	60.000	31.072
23	60.000	32.808	9.747	11.859	12.879	60.000	31.215
24	59.879	31.253	9.758	11.848	12.606	60.000	30.891
25	60.000	28.707	9.707	11.838	12.545	60.000	30.466
26	59.960	30.414	9.727	11.687	12.242	60.000	30.672
27	58.727	31.465	9.596	12.131	12.444	60.000	30.727
28	58.828	31.162	9.758	12.364	12.586	60.000	30.783
29	59.242	30.040	9.505	12.081	12.495	60.000	30.561
30	59.535	31.869	9.343	12.273	12.596	60.000	30.936
all	59.755	39.020	14.193	14.969	15.772	59.102	33.802

APPENDIX 2
VARIATION ON ONEMAX

Optimum fitness in 60 generations.
Rows: fitness scaling factor.
Columns: mutation probability.

	0.2	0.1	p_m^*	0.011	0.0091	0.0	all
1	0.07767	0.06712	0.06988	0.05368	0.05033	0.04129	0.06000
2	0.09585	0.13949	0.34156	0.50505	0.50505	0.26010	0.30785
3	0.12554	0.24698	0.52020	0.51010	0.51515	0.27200	0.36500
4	0.15064	0.33238	0.50505	0.51010	0.51010	0.27431	0.38043
5	0.17527	0.38295	0.50505	0.51010	0.51515	0.24527	0.38897
6	0.19228	0.40796	0.50505	0.50505	0.50505	0.22776	0.39053
7	0.19317	0.42202	0.51010	0.52020	0.52525	0.21614	0.39782
8	0.20096	0.42962	0.50505	0.52020	0.52020	0.21286	0.39815
9	0.20757	0.44228	0.51010	0.51515	0.52020	0.19155	0.39781
10	0.21729	0.43760	0.53535	0.52525	0.53535	0.18612	0.40616
11	0.22415	0.45142	0.53535	0.52020	0.52525	0.18860	0.40750
12	0.22137	0.47170	0.53030	0.52020	0.52525	0.18232	0.40852
13	0.21696	0.44663	0.53030	0.52020	0.53030	0.17638	0.40346
14	0.21929	0.45593	0.52525	0.52525	0.53030	0.19143	0.40791
15	0.21923	0.45895	0.52020	0.52020	0.53030	0.18281	0.40528
16	0.22790	0.46893	0.52525	0.51515	0.52525	0.17682	0.40655
17	0.22639	0.47022	0.52525	0.52525	0.53030	0.17476	0.40870
18	0.21992	0.46495	0.52525	0.52525	0.53030	0.16417	0.40497
19	0.22796	0.46760	0.53030	0.52020	0.53030	0.15905	0.40590
20	0.22251	0.46760	0.53535	0.52020	0.52525	0.16337	0.40571
21	0.22409	0.45987	0.53535	0.52525	0.53030	0.16288	0.40629
22	0.22860	0.46495	0.53535	0.52525	0.53030	0.16516	0.40827
23	0.22941	0.48221	0.53030	0.52525	0.53030	0.17055	0.41134
24	0.23153	0.46889	0.53030	0.53030	0.53030	0.17132	0.41044
25	0.23166	0.47557	0.53030	0.53030	0.53030	0.16514	0.41055
26	0.23625	0.47026	0.53030	0.52525	0.52525	0.16806	0.40923
27	0.23991	0.46893	0.53030	0.52525	0.53030	0.16687	0.41026
28	0.23145	0.46893	0.53030	0.52525	0.53030	0.16469	0.40849
29	0.23152	0.47026	0.52525	0.52525	0.53535	0.16949	0.40952
30	0.22725	0.47557	0.52525	0.52525	0.53030	0.16060	0.40737
all	0.20579	0.42126	0.50311	0.50499	0.50959	0.18506	0.38830

APPENDIX 3
F6

Optimum fitness in 60 generations.
Rows: fitness scaling factor.
Columns: mutation probability.

	0.2	0.1	p_m^*	0.011	0.0091	0.0	all
1	0.9959	0.9959	0.9947	0.9886	0.9887	0.9794	0.9905
2	0.9962	0.9958	0.9924	0.9798	0.9785	0.9729	0.9859
3	0.9958	0.9970	0.9913	0.9779	0.9761	0.9706	0.9848
4	0.9963	0.9970	0.9918	0.9824	0.9794	0.9710	0.9863
5	0.9966	0.9965	0.9923	0.9806	0.9756	0.9687	0.9851
6	0.9964	0.9963	0.9909	0.9809	0.9802	0.9709	0.9859
7	0.9967	0.9969	0.9898	0.9794	0.9775	0.9680	0.9847
8	0.9966	0.9967	0.9925	0.9815	0.9769	0.9700	0.9857
9	0.9966	0.9969	0.9916	0.9791	0.9779	0.9685	0.9851
10	0.9967	0.9970	0.9905	0.9781	0.9775	0.9684	0.9847
11	0.9968	0.9966	0.9912	0.9795	0.9775	0.9681	0.9849
12	0.9970	0.9966	0.9921	0.9808	0.9759	0.9688	0.9852
13	0.9965	0.9966	0.9922	0.9808	0.9773	0.9688	0.9854
14	0.9965	0.9967	0.9920	0.9797	0.9760	0.9659	0.9845
15	0.9968	0.9965	0.9915	0.9808	0.9769	0.9661	0.9848
16	0.9966	0.9969	0.9920	0.9810	0.9756	0.9647	0.9845
17	0.9966	0.9969	0.9912	0.9823	0.9749	0.9660	0.9847
18	0.9968	0.9968	0.9934	0.9805	0.9760	0.9660	0.9849
19	0.9966	0.9967	0.9913	0.9799	0.9769	0.9660	0.9846
20	0.9968	0.9968	0.9909	0.9797	0.9774	0.9655	0.9845
21	0.9967	0.9962	0.9903	0.9800	0.9765	0.9677	0.9845
22	0.9964	0.9966	0.9890	0.9798	0.9776	0.9678	0.9845
23	0.9968	0.9967	0.9913	0.9780	0.9775	0.9658	0.9843
24	0.9965	0.9969	0.9916	0.9802	0.9780	0.9655	0.9848
25	0.9969	0.9969	0.9917	0.9810	0.9779	0.9650	0.9849
26	0.9971	0.9970	0.9921	0.9806	0.9780	0.9655	0.9851
27	0.9969	0.9968	0.9902	0.9807	0.9777	0.9647	0.9845
28	0.9966	0.9963	0.9898	0.9794	0.9777	0.9658	0.9843
29	0.9966	0.9965	0.9905	0.9794	0.9776	0.9661	0.9844
30	0.9967	0.9964	0.9915	0.9802	0.9773	0.9666	0.9848
all	0.9966	0.9967	0.9915	0.9804	0.9776	0.9678	0.9851

APPENDIX 4
SF6

Optimum fitness in 60 generations.
Rows: fitness scaling factor.
Columns: mutation probability.

	0.2	0.1	p_m^*	0.011	0.0091	0.0	all
1	0.9956	0.9960	0.9946	0.9880	0.9875	0.9816	0.9905
2	0.9959	0.9965	0.9967	0.9849	0.9861	0.9802	0.9900
3	0.9963	0.9972	0.9971	0.9844	0.9841	0.9790	0.9897
4	0.9966	0.9969	0.9967	0.9850	0.9868	0.9786	0.9901
5	0.9965	0.9972	0.9971	0.9865	0.9861	0.9780	0.9902
6	0.9968	0.9974	0.9966	0.9867	0.9869	0.9757	0.9900
7	0.9970	0.9974	0.9955	0.9864	0.9880	0.9763	0.9901
8	0.9969	0.9973	0.9971	0.9862	0.9859	0.9761	0.9899
9	0.9967	0.9974	0.9967	0.9858	0.9867	0.9780	0.9902
10	0.9969	0.9974	0.9965	0.9879	0.9873	0.9733	0.9899
11	0.9967	0.9974	0.9963	0.9856	0.9851	0.9740	0.9892
12	0.9965	0.9973	0.9970	0.9843	0.9863	0.9758	0.9895
13	0.9969	0.9971	0.9967	0.9861	0.9834	0.9775	0.9896
14	0.9970	0.9974	0.9965	0.9860	0.9852	0.9777	0.9900
15	0.9969	0.9973	0.9962	0.9851	0.9839	0.9784	0.9896
16	0.9969	0.9973	0.9963	0.9843	0.9843	0.9785	0.9896
17	0.9968	0.9975	0.9960	0.9862	0.9849	0.9786	0.9900
18	0.9971	0.9974	0.9964	0.9853	0.9835	0.9786	0.9897
19	0.9971	0.9974	0.9962	0.9849	0.9848	0.9789	0.9899
20	0.9970	0.9975	0.9952	0.9851	0.9843	0.9769	0.9893
21	0.9969	0.9974	0.9962	0.9861	0.9863	0.9765	0.9899
22	0.9969	0.9972	0.9957	0.9850	0.9857	0.9780	0.9897
23	0.9966	0.9973	0.9962	0.9871	0.9869	0.9785	0.9904
24	0.9968	0.9974	0.9970	0.9867	0.9861	0.9793	0.9906
25	0.9970	0.9974	0.9963	0.9876	0.9866	0.9784	0.9906
26	0.9970	0.9974	0.9960	0.9874	0.9865	0.9771	0.9903
27	0.9968	0.9974	0.9963	0.9867	0.9867	0.9766	0.9901
28	0.9968	0.9973	0.9962	0.9872	0.9869	0.9772	0.9903
29	0.9968	0.9973	0.9962	0.9866	0.9861	0.9774	0.9901
30	0.9968	0.9974	0.9964	0.9871	0.9852	0.9773	0.9900
all	0.9968	0.9973	0.9963	0.9861	0.9858	0.9776	0.9900