ELSEVIER SCIENCE
IRELAND

BioSystems 32 (1994) 171–182

BioSystems

On the effectiveness of crossover in simulated evolutionary optimization

David B. Fogel[a]*, Lauren C. Stayton[b]

[a]*Natural Selection, Inc., 1591 Calle De Cinco, La Jolla, CA 92037, USA*
[b]*Loral Conic, Engineering Department, 9020 Balboa Ave., San Diego, CA 92123, USA*

(Received 4 May 1993; revision received 14 July 1993)

Abstract

There has been renewed interest in using simulated evolution to address difficult optimization problems. These simulations can be divided into two groups: (1) those that model chromosomes and emphasize genetic operators; and (2) those that model individuals or populations and emphasize the adaptation and diversity of behavior. Recent claims have suggested that genetic models using recombination operators, specifically crossover, are typically more efficient and effective at function optimization than behavioral models that rely solely on mutation. These claims are assessed empirically on a broad range of response surfaces.

Key words: Evolutionary programming; Genetic algorithms; Optimization; Evolution strategies

1. Introduction

Real-valued non-linear function minimization is a recurring important problem in the biological and engineering sciences. Classic methods of minimization involve the use of gradient descent (Schwefel, 1981, pp. 20–86). A point is selected and the slope and higher-order derivatives of the function at that point are calculated to determine the direction in which to proceed. These methods are fast to converge but often stagnate at local optima. Stochastic optimization techniques, such as simulated annealing (Kirkpatrick et al., 1983), can avoid stalling at suboptimal solutions by incorporating probabilistic mechanisms for escaping from local optima. But such methods require an extrinsic temperature schedule that is analogous to the rate in which a thermodynamic system is cooled. If the system is cooled too rapidly, annealing is subject to stagnation in local minima (it degenerates into a pure descent method). If the system is cooled too slowly the method becomes impractical for generating suitable solutions to complex problems within reasonable computational limits.

* Corresponding author.

0303-2647/94/$07.00
SSDI 0303-2647(93)01404-H

An alternative stochastic optimization technique that has gained recent attention is based on simulated evolution and termed the *genetic algorithm*. As offered in Holland (1975), this procedure: (1) maintains a population of trial solutions, rather than a single point; (2) represents solutions as strings of bits {0,1} — the degree of precision required determines the length of the string; (3) generates multiple copies of existing solutions in proportion to their relative fitness; and (4) explores new possible solutions by applying genetic operators, in particular crossover and bit mutation, to existing solutions. Genetic algorithms place primary importance on modeling observed genetic mechanisms. Solutions are regarded as *chromosomes* with alternative positions along the string being termed *genes* and the various values that can occupy a position called *alleles*.

Genetic algorithms differ from other *evolutionary algorithms* that place primary emphasis on the culling effects of selection acting on continuously diverse phenotypes (Schwefel, 1981; Fogel, 1991). In these simulations, the components of the coding strings are often viewed as behavioral traits, not as specific genes, and may take on real values. Due to the pervasive nature of pleiotropy and polygeny, even single point mutations in natural genetical systems can affect multiple phenotypic traits. The behavioral change between parent and offspring populations normally follows a Gaussian distribution and this is simulated by imposing Gaussian perturbations to all components of every vector in the population. The number of offspring per parent may vary but there is no requirement for reproducing in proportion to relative fitness. A selection criterion is used to determine which trial solutions to maintain as parents for future generations.

It has been suggested that evolutionary algorithms, such as evolutionary programming (Fogel et al., 1966), that rely solely on mutation in the absence of genetic operators (e.g., crossover) are equivalent to an enumerative search, doomed to failure on all but the simplest problems (Holland, 1975, p. 16; Goldberg, 1989, p. 106). It is often asserted that: (1) genetic algorithms are better at function optimization than other evolutionary algorithms relying solely on mutation (Davis, 1991, pp. 17–18; Holland, 1992); and (2) the incorporation of crossover is the fundamental reason for this advantage (Goldberg, 1989, p. 106; Davis, 1991, p. 18; Holland, 1992). The validity of these claims can be assessed by conducting experiments on test suites of functions and comparing the effectiveness of genetic algorithms to evolutionary algorithms.

2. Method

To find the minimum of an arbitrary objective function (adaptive landscape), the classic genetic algorithm (Holland, 1975) requires variable parameters to be coded as finite length binary strings. An initial population of strings is taken at random with each component of every string having an equal chance of being set to zero or one. These 'parents' are scored with respect to the selected objective function. Copies of the parents are created probabilistically in proportion to their relative fitness. New bit strings are generated by applying crossover and bit mutation to the copies (see Fig. 1). Primary emphasis is placed on crossover; bit mutation is viewed solely as a background operator. The probabilities for crossover and bit mutation typically range over 0.6–0.95, and 0.001–0.01, respectively. This procedure is iterated until a suitable solution is discovered, or the available computational effort is exhausted. Further details regarding the specific implementation of genetic algorithms can be found in Goldberg (1989), Davis (1991), and others, and will not be duplicated here.

One limitation of the standard genetic approach is that the range and precision of each parameter must be selected a priori. Recent research by Schraudolph and Belew (1992) utilized the GAucsd/GENESIS software package to provide a comparison between optimization using the standard genetic algorithm and a new method of iteratively narrowing the genetic search space for each parameter termed *dynamic parameter encoding* (DPE) (see Fig. 2). These efforts involved a benchmark test suite of five real-valued functions (f1–f5) offered in De Jong (1975). All of the functions are detailed in Table 1 and two-dimensional views are offered in Fig. 3. The specific parameters

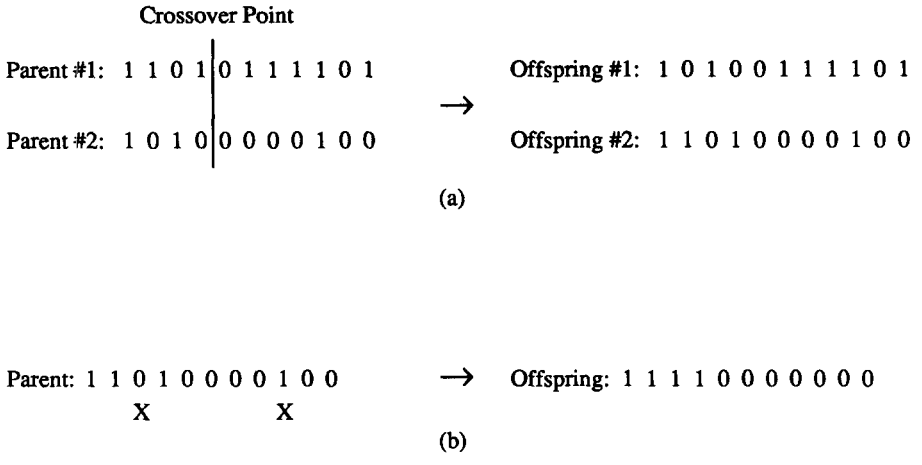


Fig. 1. The genetic operators of (a) crossover and (b) bit mutation. Crossover operates on two parents by selecting a random position and splicing the first section of each parent with the second section of the other. Bit mutation offers the possibility of randomly altering each position along the coding. In the example, 'X' denotes the position of a mutation.

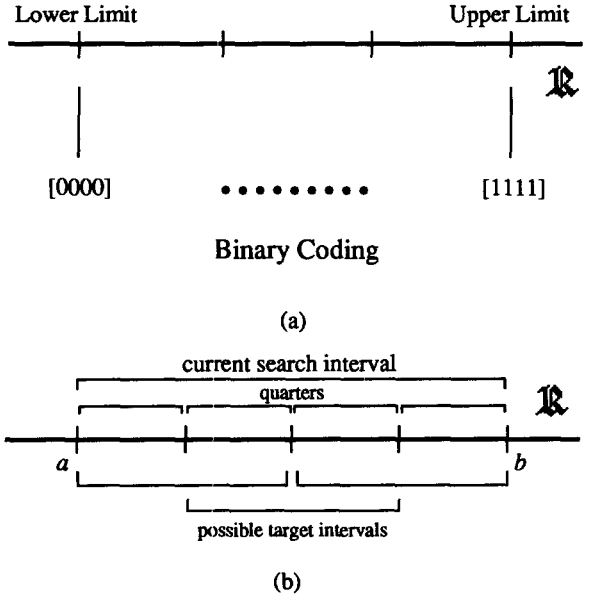


Fig. 2. Using binary coding to represent real values. A range for each parameter is selected. The string [0.0] is mapped to the lower limit and the string [1.1] is mapped to the upper limit Gray coding is typically used to represent intermediate values. (a) In standard genetic algorithms, the degree of precision required determines the number of bits to be used. (b) Under dynamic parameter encoding (DPE), the range of the parameter is modified during the execution. This provides a method for fine tuning the search. See Schraudolph and Belew (1992) for further details regarding specifics of DPE.

for the genetic algorithms studied in Schraudolph and Belew (1992) are presented in Table 2.

For comparisons, evolutionary programming (Fogel, 1992) was tested on the same set of five functions using the same parameter settings for the population size, initial range of parameters, and number of function evaluations. Specifically, the procedure is implemented as follows:

(1) Select an initial parent population of P trial solutions, x_i, $i = 1, \ldots, P$, by sampling from a uniform distribution across the preselected range of each parameter.

(2) Score all parent solutions x_i, $i = 1, \ldots, P$, with respect to the chosen function, $F(x)$.

(3) From each parent x_i, $i = 1, \ldots, P$, create an offspring, denoted as x_{i+P} by adding a Gaussian random variable with zero mean and variance equal to $F(x_i)/n^2$, to each component of the parent, where $F(x_i)$ is the parent's error score and n is the dimension of the optimization problem.

(4) Score all offspring, x_i, $i = P + 1, \ldots, 2P$, with respect to the chosen function, $F(x)$.

(5) For each x_i, $i = 1, \ldots, 2P$, select c competitors at random from the population. Conduct pairwise comparisons between x_i and each of the competitors. If the error score of x_i is less than or equal to that of its opponent, assign it a win.

(6) Select the P solutions that have the greatest

Table 1
The five test functions from De Jong (1975). x is a vector of components x_i, $[x_i]$ is the greatest integer less than or equal to x_i and $N(0,1)$ represents a normally distributed random variable with zero mean and variance of one

Function	Parameter Range
$f1{:}F(x) = \sum_{i=1}^{3} x_i^2$	[−5.12, 5.12]
$f2{:}F(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	[−2.048, 2.048]
$f3{:}F(x) = \sum_{i=1}^{5} [x_i]$	[−5.12, 5.12]
$f4{:}F(x) = \sum_{i=1}^{30} ix_i^4 + N(0,1)$	[−1.28, 1.28]
$f5{:}F(x)^{-1} = 1/K + \sum_{j=1}^{25} f_j(x)^{-1}$, $f_j(x) = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6$ where $K = 500$, $c_j = j$, and $[a_{ij}] = \begin{bmatrix} -32, -16, 0, 16, 32, -32, -16, \ldots, 0, 16, 32 \\ -32, -32, -32, -32, -32, -16, -16, \ldots, 32, 32, 32 \end{bmatrix}$	[−65.536, 65.536]

number of wins to be parents for the next generation.

(7) If the available computing time has expired then halt, else proceed to step 3.

For the current simulations, $P = 30$ and the number of competitions, c, was arbitrarily set to 10. This value represents the stringency of competition. Selection based on the number of wins provides for probabilistic survival. There are other methods for accomplishing this and no claim is made with respect to the computational efficiency of the procedure.

While the population was initialized at random over the parameter ranges noted in Table 1, solutions were allowed to vary outside these ranges in the evolutionary programming experiments. There was no requirement for the global minimum to be contained in the initial search area. One exception to this was required for f3, which continues stepping to successively lower values as the parameters tend toward negative infinity. For this problem, if any parameter of a new trial solution exceeded its initial limits, it was set equal to the limit it exceeded. For f4, the noisy quartic, all trial solutions were reevaluated every generation. In the event that a solution's score was negative, the standard deviation for generating new trials from this solution was set equal to the small positive constant 10^{-3}.

Schwefel (1981, p. 110) indicates that if the function $F(x)$ to be minimized is a simple quadratic:

$$F(x) = \sum_{i=1}^{n} x_i^2,$$

where x_i is the ith component of the n-dimensional vector x, the rate of convergence under Gaussian perturbations is optimized when the variance is proportional to $F(x)/n^2$. This procedure

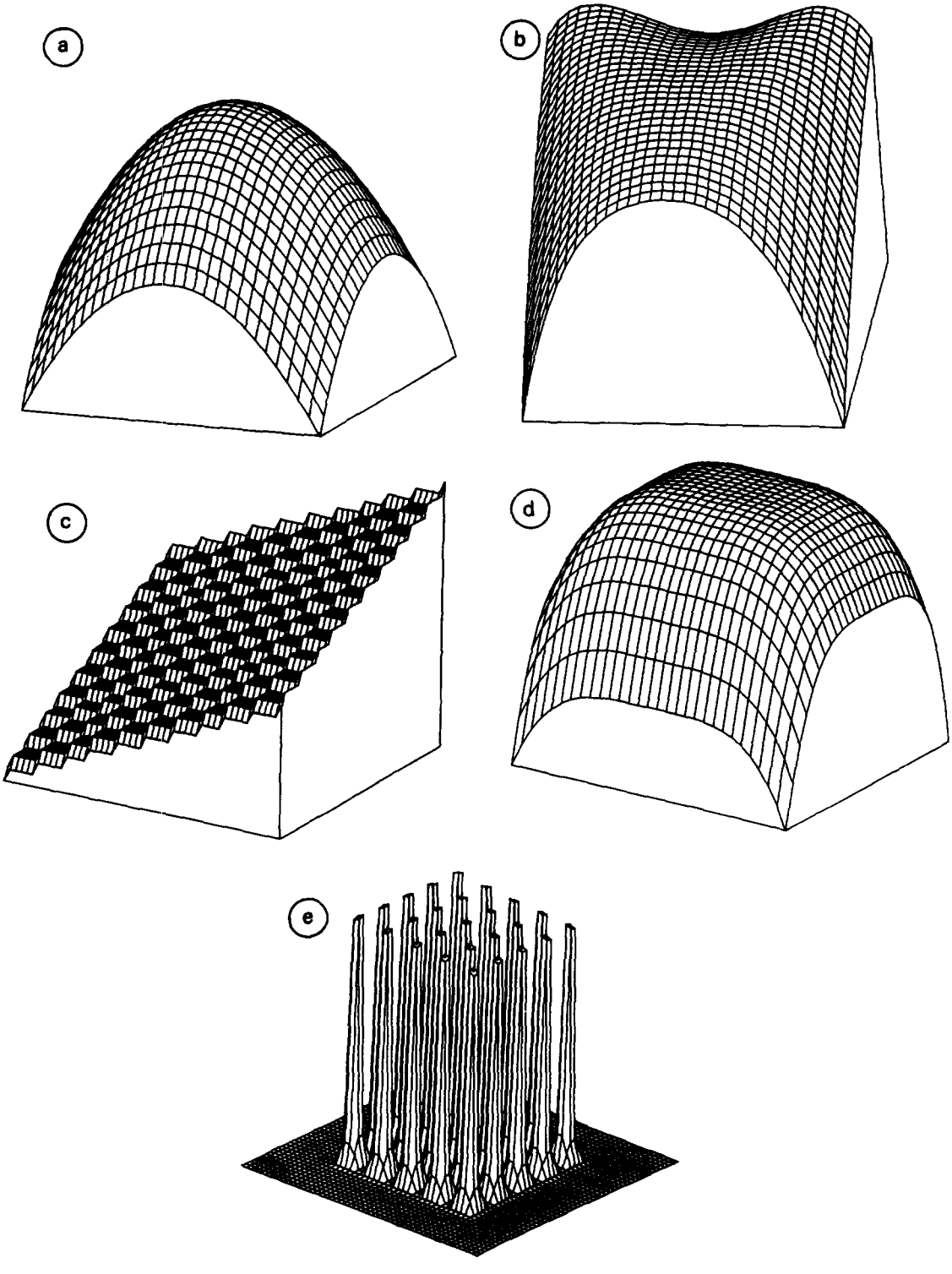


Fig. 3. Two-dimensional views of functions f1–f5 (a–e). All functions except f3 are inverted to illustrate their form more easily.

Table 2
The specific parameters used in genetic algorithm studies by Schraudolph and Belew (1992). The table indicates the bit length for each parameter in each function under the standard genetic algorithm. Gray coding was used. Three bits were used in all of dynamic parameter encoding experiments. The population size = 30 (parents), crossover rate = 0.95 and mutation rate = 0.005. See Schraudolph and Belew (1992) for other incidental settings

Function	Dimension	Bit length
f1	3	10
f2	2	12
f3	5	10
f4	30	8
f5	2	17

may not be optimal when applied to other functions. More sophisticated evolutionary methods have been proposed that involve self-adapting the variances for each parameter as well as the covariances between parameters (Schwefel, 1981, p. 145; Fogel et al., 1992). These methods do not depend on the height of the response surface. Nevertheless, for the current comparison, the above method is sufficient. As implemented, the evolutionary programming procedure is similar to evolution strategies offered by Schwefel (1981, pp. 87–154) except for the inclusion of probabilistic survival.

Of the five test functions, only the fifth contains multiple local optima. This function has been criticized as being pathologic. To provide additional tests with more reasonable surfaces that possess local optima, three functions (Fig. 4) were taken from Bohachevsky et al. (1986):

$$\text{f6: } F(x,y) = x^2 + 2y^2 - 0.3\cos(3\pi x) - 0.4\cos(4\pi y) + 0.7,$$

$$\text{f7: } F(x,y) = x^2 + 2y^2 - 0.3(\cos(3\pi x)\cos(4\pi y)) + 0.3,$$

and

$$\text{f8: } F(x,y) = x^2 + 2y^2 - 0.3(\cos(3\pi x) + \cos(4\pi y)) + 0.3.$$

Schraudolph (pers. commun.) provided the GAucsd/GENESIS software and results were obtained on f6–f8 with and without DPE. Ten trials (different initial random seed) were conducted with each method on each function. For standard genetic algorithm experiments the available range for each parameter was [−50,50] with a coding length of 14 bits. Experiments with DPE used codings of three bits, following Schraudolph and Belew (1992). Five hundred trials were conducted with evolutionary programming on each function f1–f8.

3. Experimental results

Schraudolph (pers. commun.) kindly provided the data from Schraudolph and Belew (1992) for comparisons. Table 3 indicates the best score in the population and the mean of all parents' scores after 10 080 function evaluations (336 generations), averaged over 500 trials on each function using evolutionary programming. Also indicated are the corresponding best and mean results from 10 trials with the genetic algorithm, both with and without DPE, from Schraudolph and Belew (1992) and the current experiments. Evolutionary programming outperforms both genetic algorithm methods in functions f1–f2 and f6–f8, and generates comparable performance on f3–f5. On function f4, genetic algorithms demonstrate more rapid initial optimization but quickly stagnate. Evolutionary programming eventually arrives at a population of solutions with lower error scores. The genetic algorithms are able to discover better solutions than evolutionary programming on f5, although the average error score of all parents is lower under evolutionary programming. Figs. 5 and 6 indicate the average best score in the population and the mean of the average of all parents' scores as a function of the number of generations for evolutionary programming and both genetic procedures.

Statistical comparisons were conducted between the three methods under the usual normality assumptions required for t-tests. Although the statistical significance of the data was often great (e.g., $P < 10^{-6}$ when comparing the mean best discovered solution using evolutionary programm-

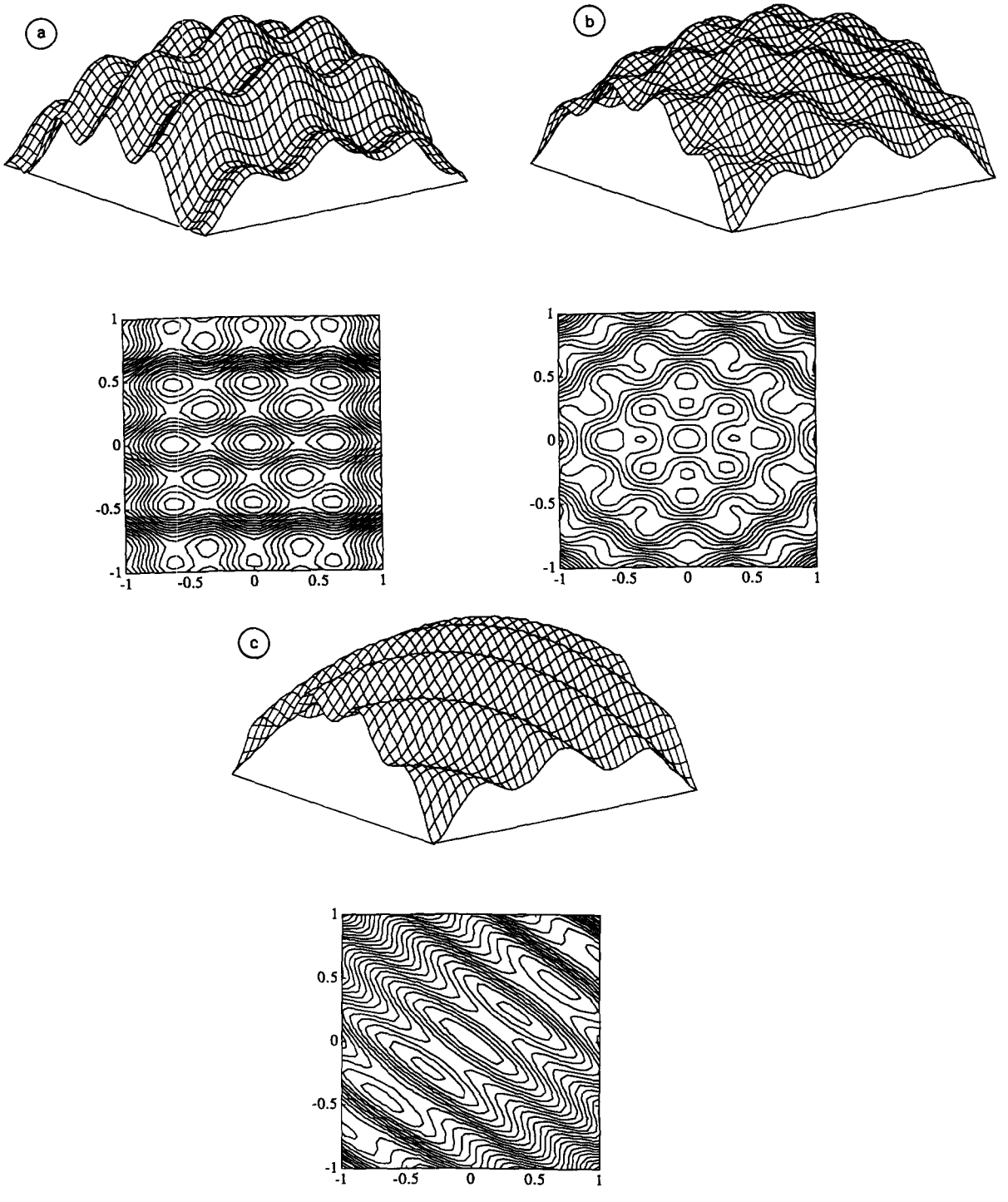


Fig. 4. Two dimensional views and contour plots of functions f6–f8 (a–c). The functions are inverted to illustrate their form more easily.

Table 3
Results of the performance of evolutionary programming and both implementations of genetic algorithms across all functions, f1–f8. The values in parentheses indicate the sample variances

f1:		f5:	
EP		EP	
Avg. Best:	3.149×10^{-66} (2.344×10^{-129})	Avg. Best:	4.168×10^{0} (9.928×10^{0})
Avg. Mean:	1.087×10^{-65} (1.794×10^{-128})	Avg. Mean:	4.194×10^{0} (1.022×10^{1})
DPE		DPE	
Avg. Best:	1.056×10^{-11} (1.072×10^{-21})	Avg. Best:	3.502×10^{0} (1.265×10^{1})
Avg. Mean:	3.618×10^{-10} (1.060×10^{-18})	Avg. Mean:	1.642×10^{1} (7.101×10^{2})
GA		GA	
Avg. Best:	2.836×10^{-4} (4.587×10^{-8})	Avg. Best:	9.980×10^{-1} (3.553×10^{-15})
Avg. Mean:	6.135×10^{-1} (1.627×10^{-1})	Avg. Mean:	1.021×10^{1} (7.165×10^{1})
f2:		f6:	
EP		EP	
Avg. Best:	1.215×10^{-14} (3.357×10^{-26})	Avg. Best:	5.193×10^{-96} (1.348×10^{-188})
Avg. Mean:	8.880×10^{-14} (2.399×10^{-24})	Avg. Mean:	9.392×10^{-94} (4.410×10^{-184})
DPE		DPE	
Avg. Best:	2.035×10^{-2} (3.315×10^{-3})	Avg. Best:	1.479×10^{-9} (1.460×10^{-18})
Avg. Mean:	8.785×10^{-2} (8.090×10^{-3})	Avg. Mean:	8.340×10^{-7} (4.649×10^{-14})
GA		GA	
Avg. Best:	2.914×10^{-2} (6.280×10^{-4})	Avg. Best:	2.629×10^{-3} (1.013×10^{-5})
Avg. Mean:	1.722×10^{0} (2.576×10^{0})	Avg. Mean:	4.022×10^{1} (6.467×10^{3})
f3:		f7:	
EP		EP	
Avg. Best:	0.0 (0.0)	Avg. Best:	8.332×10^{-101} (3.449×10^{-198})
Avg. Mean:	0.0 (0.0)	Avg. Mean:	2.495×10^{-99} (3.095×10^{-195})
DPE		DPE	
Avg. Best:	0.0 (0.0)	Avg. Best:	2.804×10^{-9} (6.831×10^{-18})
Avg. Mean:	0.0 (0.0)	Avg. Mean:	6.520×10^{-7} (7.868×10^{-14})
GA		GA	
Avg. Best:	0.0 (0.0)	Avg. Best:	4.781×10^{-3} (2.146×10^{-5})
Avg. Mean:	1.307×10^{0} (1.172×10^{-1})	Avg. Mean:	3.541×10^{1} (2.922×10^{3})
f4:		f8:	
EP		EP	
Avg. Best:	-2.575×10^{0} (7.880×10^{-1})	Avg. Best:	1.366×10^{-105} (4.479×10^{-208})
Avg. Mean:	-4.274×10^{-1} (3.206×10^{-2})	Avg. Mean:	3.031×10^{-103} (2.122×10^{-203})
DPE		DPE	
Avg. Best:	-2.980×10^{0} (1.009×10^{-1})	Avg. Best:	1.215×10^{-5} (5.176×10^{-10})
Avg. Mean:	4.704×10^{-1} (1.283×10^{-1})	Avg. Mean:	3.764×10^{-1} (9.145×10^{-1})
GA		GA	
Avg. Best:	-4.599×10^{-1} (4.265×10^{-1})	Avg. Best:	2.444×10^{-3} (4.511×10^{-5})
Avg. Mean:	1.312×10^{-1} (2.941×10^{0})	Avg. Mean:	2.788×10^{1} (1.368×10^{3})

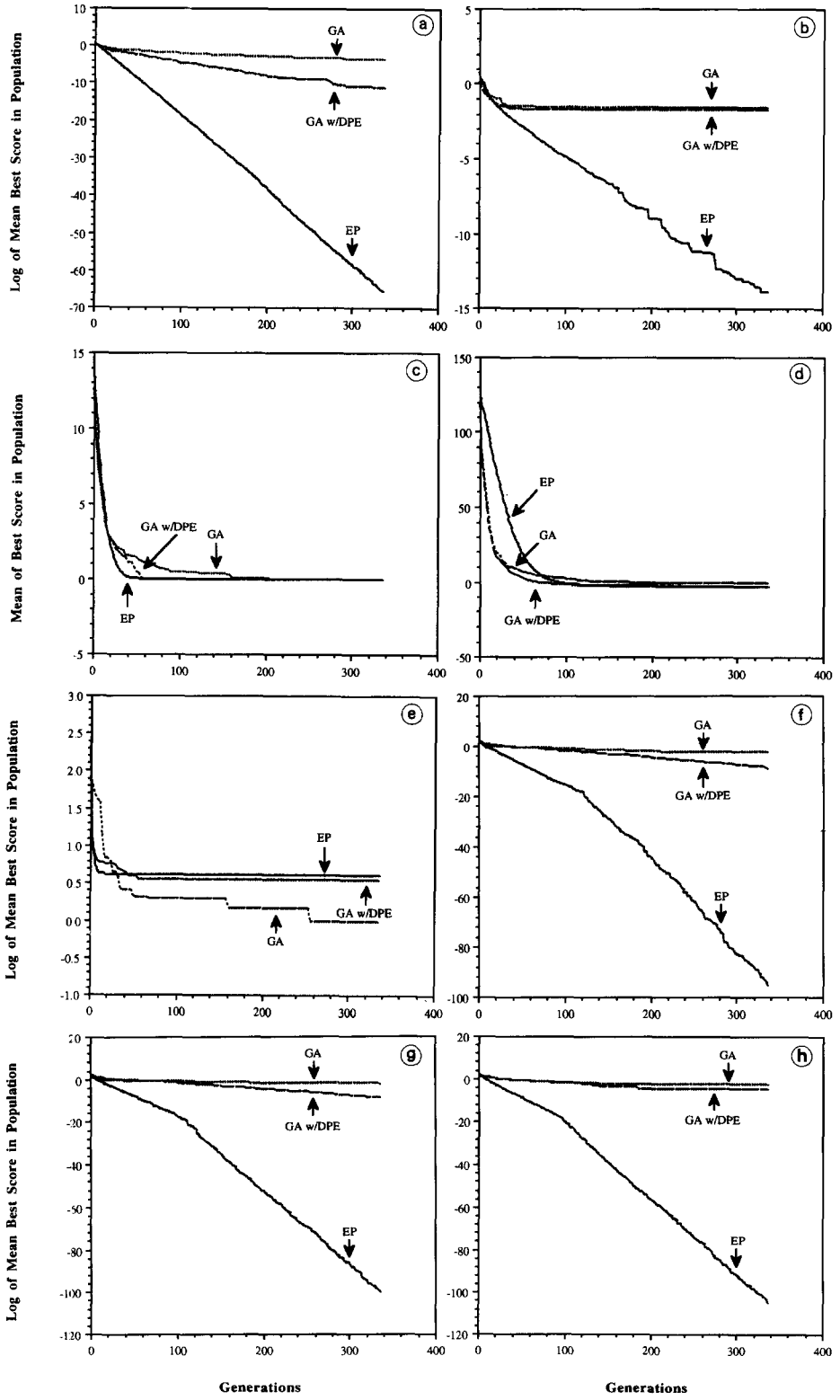
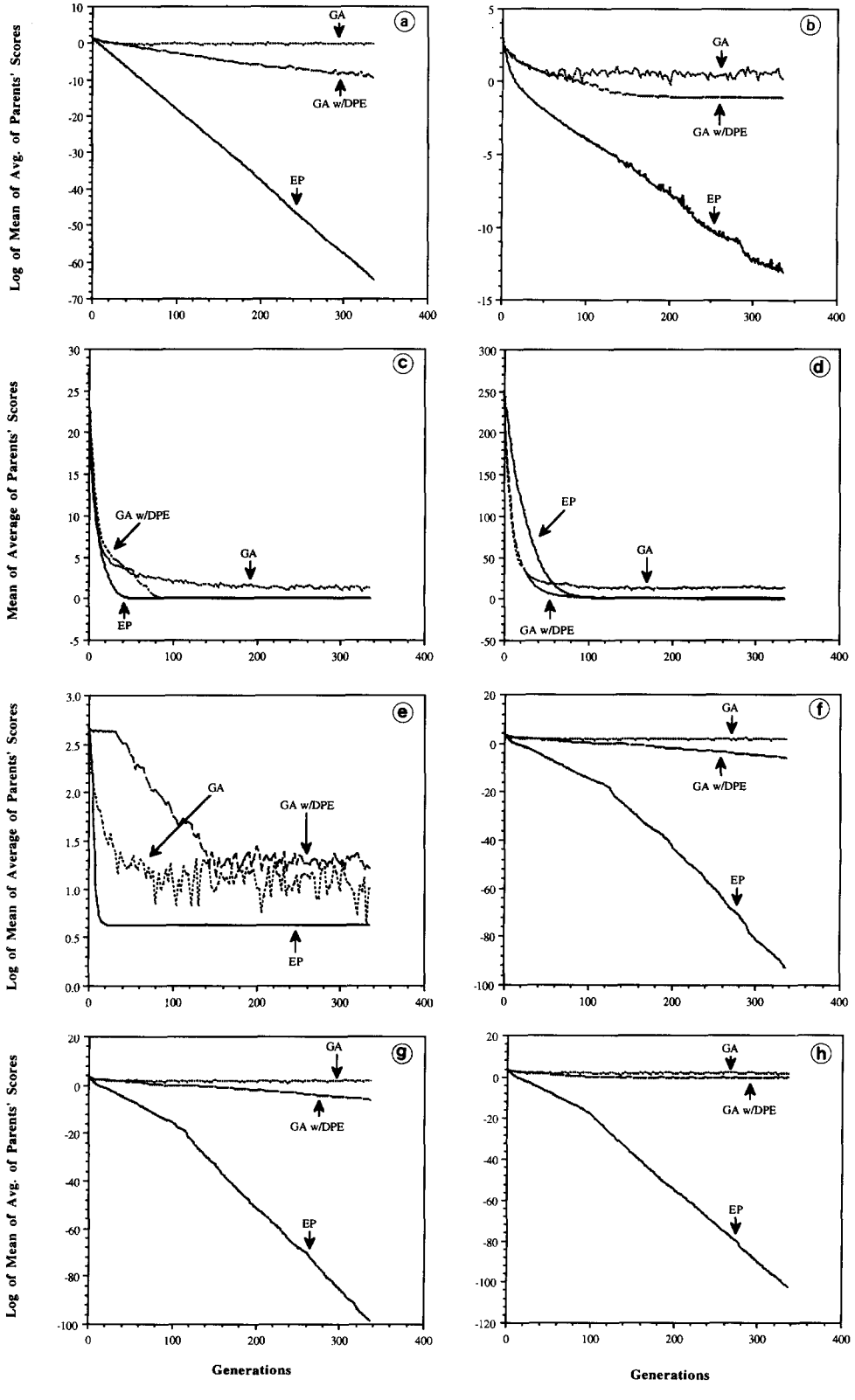


Fig. 5. The mean results of all 500 trials with evolutionary programming and 10 trials with genetic algorithms (with and without DPE) for the best solution in the population as a function of the number of generations for each function f1–f8. All trials were conducted over 336 generations (10 080 function evaluations). The results for f1–f5 are taken from Schraudolph and Belew (1992). (a) f1, (b) f2, (c) f3, (d) f4, (e) f5, (f) f6, (g) f7, (h) f8.


Log of Mean of Avg. of Parents' Scores
Mean of Average of Parents' Scores
Log of Mean of Average of Parents' Scores
Log of Mean of Avg. of Parents' Scores
GA
GA w/DPE
EP
a
b
c
d
e
f
g
h
Generations
Generations

ing to the standard genetic algorithm on f1), in many cases the data did not provide statistically significant evidence for a difference in the expected outcome between evolutionary programming and the genetic algorithm using DPE (e.g., $P < 0.2$ comparing the mean best discovered solution on f1, despite evolutionary programming outperforming the genetic algorithm with DPE by 55 orders of magnitude). This was due mainly to the small sample size provided by Schraudolph and Belew (1992) and the relatively large variance associated with their results.

An initial attempt was made to overcome this by increasing the sample size of GAucsd/GENESIS on f1 with DPE. But after 50 trials it was clear that this was going to be an unproductive effort as statistical outliers resulting from premature convergence appeared in the data. The variability of the sample was deemed largely dependent on the number of trials that prematurely converge. This appeared to occur with low probability and therefore would require numerous samples in order to provide a reasonably accurate estimate of the true variance. Thus, usual formulae for estimating the required sample size were not practical because they in part rely on accurate estimates of the true variance. It was impossible to project with confidence how many samples were necessary to discern a statistically significant difference in performance before conducting the experiment. As these conditions were observed for the simplest function (f1) that possesses a single global minimum and no local minima, it was inferred that the variability would be even worse on multi-modal functions. Further statistical analysis of the data was abandoned.

4. Discussion

It has been claimed that crossover is crucial for successful optimization because the genetic algorithm does not usually perform as well without it (Michalewicz et al., 1992). But in the majority of genetic algorithm research, initial populations are drawn at random from a large range of possible solutions. Most of these random selections provide poor performance. Simple mutation typically offers a search with a small step size. By removing crossover, the search is concentrated in each parent's relatively poor local neighborhood and relies on a succession of small steps to provide the required improvements. Crossover is effective simply because it provides a large initial step size.

Under this explanation, as the rate of crossover is usually set at 100 or more times greater than the rate of bit mutation in genetic algorithms, it should be expected that such procedures would be relatively poor at fine-tuned search, that the effectiveness of crossover would diminish as optimization proceeds, and that the procedure would tend to stagnate at suboptimal solutions prematurely. These exact conditions are often reported (e.g., Grefenstette, 1987; Davis, 1991, pp. 25–26; Michalewicz et al., 1992) and some researchers have suggested that gradient methods be applied to the final results of genetic algorithms in order to ensure convergence to at least a local optimum (Davis, 1991, p. 26).

The experimental evidence suggests that recombination, and in particular crossover, is not required for effective optimization in simulated evolution. Evolutionary algorithms, which emphasize changes at the phenotype, consistently outperformed or generated results that were comparable to genetic algorithms which emphasize genotypic effects. The results are similar to other comparisons (Fogel and Atmar, 1990; Fogel, 1992; Rizki et al., 1993). Simulating evolution as the adaptation and diversity of behaviors over successive generations of parents and offspring provides a robust method of stochastic optimization.

5. Acknowledgments

The authors are grateful to W. Atmar, T. Bäck, G.B. Fogel and L.J. Fogel for their comments and suggestions. The authors would also like to thank T. Bäck for providing independent verification of the evolutionary programming results on functions f1, f2, f4 and f5. Special thanks are owed to

Fig. 6. The mean results of all 500 trials with evolutionary programming and 10 trials with genetic algorithms (with and without DPE) for the average fitness of all parents in the population as a function of the number of generations for each function f1–f8. All trials were conducted over 336 generations (10 080 function evaluations). The results for f1–f5 are taken from Schraudolph and Belew (1992). (a) f1, (b) f2, (c) f3, (d) f4, (e) f5, (f) f6, (g) f7, (h) f8.

N.N. Schraudolph for providing the data from Schraudolph and Belew (1992), for discussions regarding the specifics of their procedure, for providing GAucsd/GENESIS software, and for the time and effort he spent reviewing our results.

6. References

Bohachevsky, I.O., Johnson, M.E. and Stein, M.L., 1986, Generalized simulated annealing for function optimization. Technometrics 28(3), 209–217.

Davis, L. (ed.), 1991, Handbook of Genetic Algorithms (Van Nostrand Reinhold, New York).

De Jong, K.A., 1975, An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Doctoral dissertation (University of Michigan, Ann Arbor).

Fogel, D.B., 1991, System Identification through Simulated Evolution: A Machine Learning Approach to Modeling (Ginn Press, Needham, MA).

Fogel, D.B., 1992, Evolving Artificial Intelligence, Doctoral dissertation (University of California at San Diego, La Jolla, CA).

Fogel, D.B., Fogel, L.J. Atmar, W. and Fogel, G.B., 1992, Hierarchic methods of evolutionary programming, in: Proc. of the First Annual Conference on Evolutionary Programming, D.B., Fogel and W. Atmar (eds.) (Evolutionary Programming Society, La Jolla, CA) pp. 175–182.

Fogel, L.J., Owens, A.J. and Walsh, M.J., 1966, Artificial Intelligence through Simulated Evolution (John Wiley & Sons, New York).

Goldberg, D.E., 1989, Genetic Algorithms in Search, Optimization and Machine Learning (Addison-Wesley, Reading, MA).

Grefenstette, J.J., 1987, Incorporating problem specific knowledge into genetic algorithms, in: Genetic Algorithms and Simulated Annealing, Davis, L. (ed.) (Morgan Kaufmann Publishers, Los Altos, CA) pp. 42–60.

Holland, J.H., 1975, Adaptation in Natural and Artificial Systems (University of Michigan Press, Ann Arbor).

Holland, J.H., 1992, Genetic algorithms, Scientific American, July, pp. 66–72.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., 1983, Optimization by simulated annealing. Science 220, 671–680.

Mayr, E., 1988, Toward a New Philosophy of Biology: Observations of an Evolutionist (Belknap Press, Cambridge, MA).

Michalewicz, Z., Janikow, C.Z. and Krawczyk, J.B., 1992, A modified genetic algorithm for optimal control problems. Comput. Math. Appl. 23(12), 83–94.

Rizki, M.M., Tamburino, L.A. and Zmuda, M.A., 1993, Evolving multi-resolution feature detectors, in: Proc. of the Second Annual Conference on Evolutionary Programming, D.B. Fogel and W. Atmar (eds.) (Evolutionary Programming Society, La Jolla, CA) pp. 108–118.

Schraudolph, N.N. and Belew, R.K., 1992, Dynamic parameter encoding for genetic algorithms. Machine Learning 9, 9–21.

Schwefel, H.-P., 1981, Numerical Optimization of Computer Models (John Wiley and Sons, Chichester).