# A Society of Hill-Climbers

M. Sebag and M. Schoenauer
Laboratoire de Mécanique des Solides and
Centre de Mathématiques Appliquées
Ecole Polytechnique
F-91128 Palaiseau Cedex – France
{Michele.Sebag,Marc.Schoenauer@polytechnique.fr}

*Abstract*— **This paper is concerned with function optimization in binary search spaces. It focuses on how hill-climbers can work together and/or use their past trials in order to speed up the search. A hill-climber is viewed as a set of mutations. The challenge is twofold: one must determine *how many bits* should be mutated, and *which bits* should preferably be mutated, or in other words, which climbing directions are to be preferred.**

**The latter question is addressed by recording the last worst trials of the hill-climbers within an individual termed *repoussoir*. The hill-climbers further explore the neighborhood of their current point as to get away from the repoussoir.**

**As to the former question, no definite answer is proposed. Nevertheless, we experimentally show that hill-climbers behave quite differently depending on whether one sets a mutation rate $p_m$ per bit, or sets the exact number $M$ of bits to mutate per individual.**

**Two algorithms describing societies of hill-climbers, with or without memory of the past trials, are described. These are experimented on several 900-bits problems, and significantly outperform standard Genetic Algorithms and Evolution Strategies.**

## I. Introduction

Most works in Evolutionary Computation (EC) explicitly refer to Darwinian evolution and biologic metaphors. The transmission of biologic information follows complex schemes, most of which seem to be designed for the species to survive changing environments [32]. However, as most applications of EC consider fixed fitness landscapes, biologic metaphors may be unnecessarily complex.

This leads to investigate other modes of transmitting information from one population to the next one. Let us distinguish three levels of information transmitted along generations:

- The basic level is that of genotypic material. The transmission of genotypic material is ensured by standard mutation and crossover operators, and genotypic material relevant to the fitness of individuals is on average transmitted proportionally to its relevance [14], [21]. However, this transmission happens to fail due to the disruptiveness of evolution operators, which leads to the "fleeting discovery" phenomenon [9]: relevant building blocks appear and disappear several times before being effectively exploited.
- The second level is that of individuals themselves: when elitist selection is used (the fittest individuals always survive, the phenomenon of fleeting discoveries is avoided. Though theoretical results in the GA framework endorse elitism [25], elitist strategy is not theoretically optimal in term of progress rate in the ES context [5]. Nevertheless, a wide range of EC algorithms use elitist selection schemes.
- The third level is that of control of evolution, intended as all means (bias) for improving the course of evolution and the transmission of relevant genotypic material. The information relative to control can be carried by individuals themselves, through genotypic yet non phenotypic-ally relevant material (e.g. type of crossover [30], [31], mask of crossover [26], mutation step size [27], [2], encoded within each individual). The information relative to control can also be stored independently from individuals; e.g. global operator rates [7], population distribution [4], directions of mutation [12], [29], beliefs about schemata of individuals [24], [6], or schemata of disruptive operators [28], [22]. In the former case, referred to as adaptive biological evolution, the control information is part of genotypic material; it is evolved and transmitted by the same mechanism.
  In the latter case, the control information is deterministically updated from the current population using exogenous heuristics: it represents an abstraction of the current (or the few past) population(s). As the evolution of an individual is here biased according to an abstraction of the other ones, this case is referred to as *sociological evolution*.

This paper focuses on sociological evolution and more precisely studies how a *society of hill-climbers* could evolve. Only binary search spaces are considered throughout the paper. In terms of genetic algorithms, a hill-climber performs a sequence of mutations.

Most works in GAs have considered small mutation rates (e.g. $10^{-3}$); and the control of binary mutation is mostly concerned with determining the average number of bits to mutate [7]. Still, mutation with strong rate can be much desirable [16]. But strong mutation asks the further question of which bits should be mutated: if it indifferently affects all bits, mutation gets intolerably disruptive as evolution goes on; this contrast with the disruptiveness of crossover decreasing as evolution goes on and population gets homogeneous.

This paper investigates how hill-climbers can exchange information in order to determine which bits should be preferably mutated, that is, which climbing directions are to be preferred in the current state of the population.

Figure 1. **Historical Society** $(\mu, \lambda, M, R, \alpha, T)$. $R = 50\%$, $\alpha = .01$, $T = 30$ *in all experiments.*

| | |
|---|---|
| **Initialization** | $x_1, \ldots, x_\mu$ are randomly selected.<br>$Repoussoir = (.5, \ldots, .5)$. |
| **Generation** | Pool = $\{x_1, \ldots, x_\mu\}$.<br>For i = 1 to $\mu$<br>   Pool = Pool $\cup$ Hill-Climb($seed\ x_i$)<br>$\{x_1, ..x_\mu\}$ = $\mu$ best individuals in Pool.<br>$y$ = average of the $R$ % worst individuals in Pool.<br>$Repoussoir = (1-\alpha) * Repoussoir + \alpha * y$. |
| **Hill-Climb**<br>($seed$, $M$,Repoussoir,$T$) | Trials = {}<br>For j = 1 to $\lambda/\mu$<br>   Do $y = seed$<br>   For k = 1 to $M$,<br>      bit $l$ = Tournament($seed, Repoussoir, T$)<br>      $y(l) = 1 - y(l)$<br>   Trials = Trials $\cup$ $y$<br>return Trials |
| **Tournament**<br>($seed$,Repoussoir,$T$) | Randomly select $T$ bits $i_1..i_T$ of the problem<br>Return $i_k$ such that $\lvert seed(i_k) - Repoussoir(i_k)\rvert$ is minimum. |

This paper investigates two kinds of hill-climber societies, respectively termed natural societies (NS) and historical societies (HS). These are presented in section 2, and briefly compared to some related works [20], [10], [11], [29], [4]. Section 3 presents and discuss an experimental validation of these schemes on six large-sized problems, and some perspectives for future research build up the final section IV.

## II. Natural and Historical Societies

This section describes two organizations for a hill-climber society. In both cases, a generation consists of several hill-climbers exploring the neighborhood of their respective current starting point, or *current seeds*. In natural societies, a hill-climber only knows its current seed. In historical societies, a hill-climber is further provided with a summary of the individuals explored in the previous generations. These schemes are compared to Evolution Strategy [27], Tabu Search [10] and Population-Based Incremental Learning [4].

### A. Natural Society

In natural societies (NS), each hill-climber explores the neighborhood of its current seed with uniform distribution. With the addition of selection at the population level, this scheme can be viewed as a variant of $(\mu + \lambda)$ Evolution Strategy (ES) [27], [2], where $\mu$ stands for the number of hill-climbers and $\lambda$ for the total number of trials allotted to the hill-climbers.

However, ESs mostly consider continuous search spaces, and use a mutation rate per coordinate: This rate depends on the coordinate and the individual in Adaptive ES [2] or is the same for all coordinates of all individuals, and is adjusted according to the 1/5th rule in Traditional ES [23]). These powerful adaptive mechanisms are, as such, ill-suited to binary spaces, which violate the strong causality principle [23]: if the mutation rate is allowed to become small enough (less than $1/N * \lambda$, if $N$ denotes the size of the problem), mutation has no effect. On the other hand, if a lower bound on the mutation rate is enforced, there is no such thing as a "very small mutation". In opposition, Gaussian mutation in continuous search spaces always has some effect as long as the mutation rate is not exactly 0.

In NS, the user sets the number $M$ of bits to mutate *per individual*; a hill-climber randomly explores the individuals differing from its seed by *exactly* $M$ bits. Rather surprisingly, and for a rather wide range of values for $M$, NS outperform Traditional ES [23] as well as Adaptive ES [20], [1] (see section III). Note however that such decision can definitely prevent the algorithm to reach the actual optimum ...

NS involves three parameters: the number $\mu$ of hill-climbers, the number $\lambda$ of trials allowed to all hill-climbers and the number $M$ of bits to mutate.

### B. Historical Society

In historical societies (HS), hill-climbers are also provided with some memory of the past generations. This memory is first designed to avoid repeating past unfruitful trials, i.e. individuals which don't pass the selection step: Indeed, if the fitness landscape is fixed, and provided the selection scheme is that of $(\mu + \lambda)$ ES, an individual that was not selected at a given generation will never be selected in the future. Further, this memory is used to determine preferred directions for hill-climbing.

How could low-fitness individuals contribute to a better exploitation of highly fit individuals ? The answer originates from a Machine Learning perspective [19], [18]. Consider individuals $x$, $y$, $z$ and $t$, where $x$ has a (comparatively) high fitness, and $y$, $z$ and $t$ all have a low fitness (Table 1).

Bit 1 takes different values for the highly fit $x$ and the low fit $y$, $z$ and $t$. By induction, this difference in their genotypes may be considered a "cause" for the difference in their fitness; that is, from these examples, one could consider the rule *If bit 1 is 0, then the fitness is high.* In consequence, when exploring the neighborhood of $x$, one should preferably mutate bit 5 (which has no visible effects whatsoever) than bit 1 (a possible cause of high fitness). More generally, the more difference a bit makes between the seed and the unfruitful trials, the less this bit should be mutated.

| | bit 1 | bit 2 | bit 3 | bit 4 | bit 5 | Fitness |
|---|---|---|---|---|---|---|
| $x$ | 0 | 0 | 0 | 0 | 0 | high |
| $y$ | 1 | 1 | 1 | 1 | 0 | low |
| $z$ | 1 | 0 | 0 | 1 | 0 | low |
| $t$ | 1 | 1 | 0 | 0 | 0 | low |

*Table 1: What can be gained by comparing low and high fit individuals ?*

Practically, the past unfruitful trials are stored as an average individual termed *repoussoir* (French for foil, or repeller), which belongs to $[0,1]^N$ if $N$ denotes the dimension of the problem. This repoussoir is updated every generation by relaxation (linear combination of average of current worse individuals and previous *repoussoir*, see Figure 1).

Each hill-climber selects the bits to mutate by means of *repoussoir*-based tournament: the winner is the bit which takes the most similar value for the seed and the repoussoir. Thereby, the offspring will be still farther from the *repoussoir* than the seed was: the climbing direction consists to get away from the *repoussoir*.

HS involves six parameters (Figure 1): besides the three parameters $\mu$, $\lambda$ and $M$ of NS, it also depends on the fraction $R$ of individuals used to construct the *repoussoir*, the relaxation factor $\alpha$, and the tournament size $T$ used to select the bits to mutate. Only $M$ was found critical (see section III).

### C. Related works

Historical hill-climber societies continue a previous work of the authors devoted to "mutation by imitation" [29]; two modifications were needed to address large sized problems. First, the repoussoir now reflects the worst individuals in all previous generations, according to the relaxation factor $\alpha$, whereas it previously depended on the last population only (e.g. $\alpha = 1$). Second, the bits to mutate were determined on the basis of a roulette wheel selection, the probability of mutating the $k$-th bit being a decreasing function of $|x(k) - Repoussoir(k)|$. It turned out that this function was quite difficult to adjust. The roulette wheel mechanism was therefore replaced by a tournament-based selection, more robust and less computationally expensive.

The idea of storing the past trials in order not to repeat them is borrowed to Tabu Search [10] and has been already explored in induction-guided evolution [22]. The first difference lies in the fact that the memory of the past search is stored as a distribution rather than as a list of previous trials [10] or schemas of individuals [22]; further, this memory is used stochastically rather than deterministically.

In the framework of continuous parameter optimization, Hansen & al. [12] use a memory made of a basis of $\mathbb{R}^n$, in which the adaptive mutation is computed. The adaptive mechanism then becomes independent of the underlying system of coordinates. This strategy, however, does not transpose immediately to binary spaces.

In the context of binary spaces, historical societies can also be compared to the Population-Based Incremental Learning (PBIL) scheme [4]. PBIL constructs an "ideal individual", summarizing the best individuals in the previous populations (exactly as the repoussoir summarizes the worst individuals in HS). This ideal individual can be viewed as a distribution; every generation, the population is constructed from scratch according to this distribution. In particular, there exists no transmission of genetic material between one population and the next one (the only transmitted information is that of the ideal individual). Therefore, there is little chance indeed to rediscover the previous best individual. This obviously is an advantage when the fitness landscape presents many local optima.

But PBIL should be adversely affected, when the region of high fitness is "narrow" and more resembles a path than a hill: if you ever leave the path you are not certain to find it again. Evidence for this is given by experimentations on the 91-bit Long Path problem [15]:

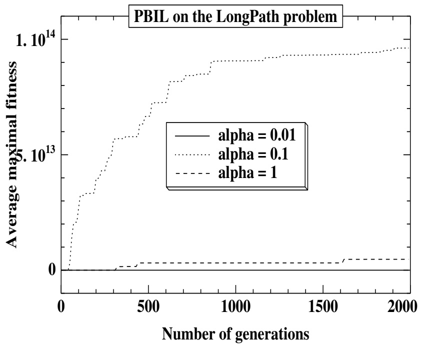


Figure 2. *PBIL on the 91-bit Long Path problem.*

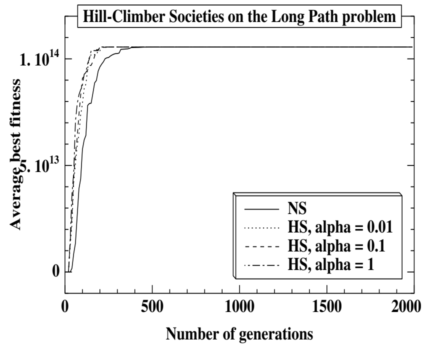


Figure 3. *HS and NS on the 91-bit Long Path problem.*

This problem admits a unique global optimum, no local optima, and was purposely designed to discourage standard hill-climbers. The fitness landscape is composed of two different regions; the first one resembles the OneMax landscape, and culminates in $x = (0,..0)$; the second region is composed of a path of exponential length, starting at $x$. The fitness of an individual either is its rank on the path, if it belongs to the path; or its number of zeros otherwise. To solve this problem, an evolutionary algorithm must have different skills, e.g. find the top of a hill (in the OneMax part of the landscape) and find shortcuts on an otherwise exponentially long path [13].

The results obtained by NS, HS and PBIL are averaged on 30 runs. Figure 2 and 3 plots the best fitness reached for a given number of generations. The four parameters of PBIL are set as in [3]; in particular the number of trials per generation is 100. To permit a fair comparison, $\lambda$ is set to 100 for NS and HS too. NS parameters finally are $\lambda = 100$, $\mu = 10$ and $M = 2$. HS parameters are $\lambda = 100$, $\mu = 1$, $M = 2$, $T = 20$ (very similar results are obtained for $T = 10$ and $T = 30$) and $R = 50\%$. The relaxation factor $\alpha$ (for HS and PBIL) varies from 1 to 0.01; it is shown to have much influence on PBIL results for the LongPath problem (Figure 2), in contrast with HS (Figure 3).

## III. Experimental results

For the sake of convenience, experimentations have considered numerical problems which have already been used to evaluate PBIL [3].

### A. Problems

All three functions involve 100 numerical variables $x_i$ coded on 9 bits each and varying in [-2.56,2.56[. Both standard binary and Gray encoding have been considered,

$$y_1 = x_1 \qquad y_i = x_i + y_{i-1},\ i \geq 2 \qquad F_1 = \frac{100}{10^{-5} + \Sigma_i |y_i|}$$

$$y_1 = x_1 \qquad y_i = x_i + sin(y_{i-1}),\ i \geq 2 \qquad F_2 = \frac{100}{10^{-5} + \Sigma_i |y_i|}$$

$$F_3 = \frac{100}{10^{-5} + \Sigma_i |.024 * (i+1) - x_i|}$$

The maximum of both $F_1$ and $F_2$ is reached at point (0,...,0) and has value $10^7$ while the maximum of function $F_3$ is reached for $x_i = 0.024 * (i+1)$. However, due to the coarse discretization (9 bits), the maximum value here for $F_3$ is 416.64.

### B. Reference algorithms

Multiple restart Hill-Climbers denoted HC1 and HC2, as well as two genetic algorithms denoted SGA and GA-scale, have been used in [3] as reference algorithms:

HC1 randomly considers the neighbors of the seed (differing from the seed by one bit); the seed is replaced by the first neighbor which strictly improves on the seed. If all neighbors of the seed have been considered, HC1 is restarted with another seed. HC2 differs from HC1 in that it allows to replace the seed by a neighbor having similar fitness.

SGA is a standard GA which uses two point crossover, with crossover rate 100%, mutation rate $10^{-3}$, population size 100 and elitist selection. GA-scale differs from SGA in two respects: it uses uniform crossover with rate 80%, and the fitness of the worst individual is subtracted from the fitness of all individuals in the population before selection. A more detailed description of these algorithms, as well as their results on the considered problems, can be found in [3].

Two additional reference algorithms have also been considered:

TES (for *Traditional evolution strategy*) is a binary $(\mu+\lambda)$-ES involving a single mutation rate per bit $\sigma$ ; $\sigma$ is modified according to the Rechenberg's 1/5 rule [23]. The geometrical factor used to increase $\sigma$ ranges from 1.1 to 2.

AES (for *Adaptive ES*) is a boolean $(\mu + \lambda)$-ES that uses the adaptive mutation of [20] as described in [1] with minor differences: Each individual is attached one mutation rate $p$ (the probability of mutation of any single bit) which itself undergoes mutation according to the following *Obalek's rule*:

$$p := \left(1 + \frac{1-p}{p}.exp(\gamma\ .\ N(0,1))\right) \qquad \text{with } \gamma = \frac{0.5}{\sqrt{2\sqrt{N}}},$$

with a lower bound of $1/N$ on $p$ to guarantee effective mutation. In both cases, $\mu$ is 10 and $\lambda$ is 100.

### C. Results

The parameters for NS (and HS) are $\mu = 10$ and $\lambda = 50$ ($R = 50\%$, $T = 30$ and $\alpha = .01$), which were found rather robust for all problems through systematical trials. On the opposite, the value of $M$ seems critical depending on the fitness landscape.

All algorithms are allowed 200,000 fitness calculations per run. Results averaged over 20 independant runs, are shown on Table 2: The results of HCi, GAs and PBIL are taken directly from [4].

NS and HS significantly outperform standard GAs and ESs, and obtain similar or better results than PBIL on all problems but F3 Binary. However, the case of F3 is to be handled separately: the easy F3 Gray is solved exactly by the Hill-Climbers HC1 and HC2, as well as by NS and HS with $M = 1$. As a consequence, F3 Binary requires jumps of variable length (e.g. in order to pass from 10....0 to 011...1), which is harder to achieve within a single fixed $M$-bit mutation than in the PBIL scheme.

Letting aside both F3 functions, these results asks for three comments. First of all, the efficiency of the simple mechanism of NS, which obtains better results than all other methods but PBIL, is surprising, but requires that $M$ is well adjusted. Moreover, NS obtains really bad results for $M = 1$ comapred to all values of $M > 1$ (results not shown on Table 2). Indeed, for $M = 1$, NS gets trapped in local minima as any standard hill-climber. But this distinction between exploiting the seed and exploring new regions of the search space vanishes as $M$ increases.

Second, HS significantly improves on NS, even with the same value of $M$ (results not shown on Table 2). But further, the *repoussoir*-based choice of the bits to mutate allows to modify more bits simultaneously in the same mutation: the best value of $M$ for HS is in most cases larger than for NS. This might be one reason for the greater overall efficiency observed in the results of Table 2.

| | HC1 | HC2 | SGA | GA-scale | AES | TES | PBIL | NS | M | HS | M |
|---|---|---|---|---|---|---|---|---|---|---|---|
| F1 binary | 1.04 | 1.01 | 1.96 | 1.72 | 2.37 | 1.87 | 2.12 | **2.87** | 3 | 2.51 | 4 |
| F1 Gray | 1.21 | 1.18 | 1.92 | 1.78 | 2.04 | 1.66 | 2.62 | 2.39 | 3 | **2.84** | 4 |
| F2 binary | 3.08 | 3.06 | 3.58 | 3.68 | 3.94 | 3.61 | 4.40 | 4.24 | 4 | **4.49** | 5 |
| F2 Gray | 4.34 | 4.38 | 3.64 | 4.63 | 5.18 | 4.66 | 5.61 | 5.98 | 2 | **8.40** | 2 |
| F3 binary | 8.07 | 8.10 | 9.17 | 12.30 | 9.06 | 10.46 | **16.43** | 12.80 | 4 | 14.12 | 5 |
| F3 Gray | **416.64** | **416.64** | 28.35 | 210.37 | 380.3 | **416.64** | 366.77 | **416.64** | 1 | **416.64** | 1 |

Table 2: *Average best fitness after 200 000 function evaluations for 6 900-bits problems. Results for the PBIL, HCi, SGA and GA are taken directly from [3].*

Last and overall, the main weakness of both NS and HS is that parameter $M$ must be supplied by the user. We therefore made some attempts for adjusting $M$ online, either at the level of individuals as in ES [27], or at the level of the population, on the basis of rewards *a la* Davis [7].

But both heuristics seemingly suffer from myopia. An option which often gets small rewards (as happens for $M = 1$) will be favored over an option which seldom gets big rewards (as happens for higher values of $M$), no matter how the cumulated rewards of both options compare. Besides, if an option is rarely selected, it likely gets no rewards, and it is still more rarely selected...
Further, complementary experiments (not reported here) show that it is preferable to adjust the mutation rate per individual, than per bit: both NS and HS obtain better results when the mutation rate is per individual (i.e. mutate exactly $M$ bits) than when the mutation rate is $M/(total\ nb\ of\ bits)$ per bit (i.e. mutate $M$ bits on average), for wide range of values of $M$.

## IV. Conclusion and Perspectives

This work focuses on the adjustment of the mutation rate and the mutation distribution, in the framework of binary "Hill-Climber Societies".

In terms of hill-climbing, the mutation rate governs the length and the variance of the jumps allowed. Experimental validation on several large-sized problems strongly suggests that one should rather fix the length of the jumps, and use a fixed number $M$ of bits to mutate per individual, than adaptively adjust a mutation rate per bit: be the adjustment based on the 1/5th or the Obalek rules (section III-B), it tends to favor small frequent steps over big rare steps[1]. Further, for optimal and near-optimal values of $M$ (the length of the jumps), a low variance is preferable: better using a number $M$ of bits to mutate per individual than a mutation rate per bit of $M/(total\ nb\ of\ bits)$.
It remains to determine the desirable value of $M$. The adjustment of $M$ based on rewards *a la* Davis [7] suffers from the same limitations as the adaptive adjustment of a mutation rate per bit. Further work is concerned with determining $M$ off line, for instance by measuring the fitness distance correlation [17].

In what regards the mutation distribution, the goal consists in determining judicious climbing directions. The originality of our approach is to reverse the question and ask where *not to go*, rather than where to go. Finding where to go is amenable to find the "gradient" of the fitness function — which is difficult in binary spaces. But finding where to go can be partially answered on the basis of the past history of the search: one should not go in the regions where unfruitful explorations have been done. This remark leads to construct a *repoussoir* summarizing the worst individuals explored in the previous generations. The preferred climbing direction thereafter consists in getting away from the repoussoir. This HS approach resembles Tabu search [10] in that it memorizes the past trials in order not to repeat them. The difference lies in the coding and use of the memory, which is deterministic in Tabu search and stochastic in HS. Many other uses of the past history of search remain to be invented; still, HS outperforms standard GAs and the non-recombinant $(\mu, \lambda)$-ESs on all 6 problems experimented so far, and outperforms the PBIL scheme on 5 out of these 6 problems.
Further work is concerned with extending HS to continuous search spaces.

[1]This is to be related to the trend of GAs, toward optimizing the average rather than the maximal fitness in the population [8].

## Acknowledgements

The authors want to thank the anonymous reviewers whose comments help to improve the readibility of this paper.

## References

[1] T. Bäck and M. Schütz. Evolution strategies for mixed-integer optimization of optical multilayer systems. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the $4^{th}$ Annual Conference on Evolutionary Programming*. MIT Press, March 1995.

[2] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[3] S. Baluja. An empirical comparizon of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University, 1995.

[4] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithms. In A. Prieditis and S. Russel, editors, *Proceedings of ICML95*, pages 38–46. Morgan Kaufmann, 1995.

[5] H.-G. Beyer. On the asymptotic behavior of multi-recombinant evolution strategies. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the $4^{th}$ Conference on Parallel Problems Solving from Nature*, LNCS 1141, pages 122–131. Springer Verlag, 1996.

[6] M.J. Cavaretta. Using a cultural algorithm to control genetic operators. In A.V. Sebald and L.J. Fogel, editors, *$3^{rd}$ Annual Conference on Evolutionary Programming*, pages 158–166. World Scientific, 1994.

[7] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the $3^{rd}$ International*

*Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, 1989.

[8] K. A. DeJong. Are genetic algorithms function optimizers ? In R. Manner and B. Manderick, editors, *Proceedings of the* $2^{nd}$ *Conference on Parallel Problems Solving from Nature*, pages 3–13. North Holland, 1992.

[9] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithms : Some anomalous results and their explanation. *Machine Learning*, pages 285–319, 1993.

[10] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.

[11] F. Glover and G. Kochenberger. Critical event tabu search for multidimensional knapsack problems. In *Proceedings of the International Conference on Metaheuristics for Optimization*, pages 113–133. Kluwer Publishing, 1995.

[12] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In L. J. Eshelman, editor, *Proceedings of the* $6^{th}$ *International Conference on Genetic Algorithms*, pages 57–64. Morgan Kaufmann, 1995.

[13] C. Höhn and C. Reeves. Are long path problems hard for genetic algorithms? In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the* $4^{th}$ *Conference on Parallel Problems Solving from Nature*, volume 1141 of *LNCS*, pages 134–143. Springer Verlag, 1996.

[14] J. H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.

[15] J. Horn and D.E. Goldberg. Genetic algorithms difficulty and the modality of fitness landscapes. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 243–269. Morgan Kaufmann, 1995.

[16] T. Jones. Crossover, macromutation and population-based search. In L. J. Eshelman, editor, *Proceedings of the* $6^{th}$ *International Conference on Genetic Algorithms*, pages 73–80. Morgan Kaufmann, 1995.

[17] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman, editor, *Proceedings of the* $6^{th}$ *International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

[18] R.S. Michalski. A theory and methodology of inductive learning. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning : an artificial intelligence approach*, volume 1, pages 83–134. Morgan Kaufmann, 1983.

[19] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[20] J. Obalek. *Rekombinationsoperatoren für Evolutionsstrategieren.* Diploma thesis, Universität Dortmund, Fachbereich Informatik, 1994.

[21] N. J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–20, 1991.

[22] C. Ravisé and M. Sebag. An advanced evolution should not repeat its past errors. In L. Saitta, editor, *Proceedings of the* $13^{th}$ *International Conference on Machine Learning*, pages 400–408, 1996.

[23] I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution.* Fromman-Holzboog Verlag, Stuttgart, 1973.

[24] R.G. Reynolds. An introduction to cultural algorithms. In *Proceedings of the* $3^{rd}$ *Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, 1994.

[25] G. Rudolph. Convergence analysis of canonical genetic algorithm. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[26] J.D. Schaffer and A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of the* $2^{nd}$ *International Conference on Genetic Algorithms*, pages 36–40. Morgan Kaufmann, 1987.

[27] H.-P. Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, New-York, 1981. 1995 – $2^{nd}$ edition.

[28] M. Sebag and M. Schoenauer. Controlling crossover through inductive learning. In Y. Davidor, H.-P. Schwefel, and R. Manner, editors, *Proceedings of the* $3^{rd}$ *Conference on Parallel Problems Solving from Nature.* Springer-Verlag, LNCS 866, 1994.

[29] M. Sebag and M. Schoenauer. Mutation by imitation in boolean evolution strategies. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the* $4^{th}$ *Conference on Parallel Problems Solving from Nature*, pages 356–365. Springer-Verlag, LNCS 1141, 1996.

[30] W. M. Spears. Adapting crossover in a genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings of the* $4^{th}$ *International Conference on Genetic Algorithms.* Morgan Kaufmann, 1991.

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

[32] C. Wills. *The wisdom of genes.* Elsevier, 1991.