

# Decision making in a hybrid genetic algorithm

Fernando G. Lobo and David E. Goldberg

**Abstract**— There are several issues that need to be taken in consideration when designing a hybrid problem solver. This paper focuses on one of them—decision making. More specifically, we address the following questions: given two different methods, how to get the most out of both of them? When should we use one and when should we use the other in order to get maximum efficiency? We present a model for hybridizing genetic algorithms (GAs) based on a concept that decision theorists call probability matching and we use it to combine an elitist selecto-recombinative GA with a simple hillclimber (HC). Tests on an easy problem with a small population size match our intuition that both GA and HC are needed to solve the problem efficiently.

## I. INTRODUCTION

It is very unlikely that a GA will outperform a specialized scheme tailored to a problem. However, a combination of the two usually performs better than either one alone. This happens because on a hybrid there is the possibility of incorporating domain knowledge, which gives it an advantage over a pure blind searcher such as a GA. That's the reason why most of the successful applications of GAs have been hybrids. An example was the EnGENEous project at General Electric [1]. They used a combination of GAs, expert systems and numerical optimization techniques to create a general purpose tool. The combination performed better than each technique when applied alone, and with it, they could improve the efficiency of the gas turbine that is part of the Boeing 777 jet engine. In [2], a number of other successful hybrid applications are described.

Although there is empirical evidence that hybridization can give a boost in performance, little theory has been developed on how to design hybrid GAs and most of the work in this area is done empirically to tune the specific problem that people are trying to solve. The theoretical work in this area has focused mainly on combining the global search power of the GA with the local search power of a HC. Two lines of work have been studied with quite a bit of extension: the Lamarckian Evolution and the Baldwin Effect. Both use the metaphor that an individual learns (hillclimbs) during its lifetime (generation). In the Lamarckian case, the resulting individual (after hillclimbing) is put back into the population. In the Baldwinian case only the fitness is changed and the genotype remains unchanged. The drawback of the Lamarckian approach is loss of diversity. On the contrary, the Baldwinian strategy maintains the diversity in the population.

F. Lobo is a PhD student at "Grupo de Análise de Sistemas Ambientais", FCT/UNL, Portugal. He is also a visiting student at the Department of General Engineering, University of Illinois at Urbana-Champaign. E-mail: fgl@mail.fct.unl.pt

D. Goldberg is a Professor in the Department of General Engineering, University of Illinois at Urbana-Champaign. E-mail: deg@uiuc.edu

The next section outlines several issues that need to be addressed on a complete theory of hybridization. Section 3 describes a model for combining two different methods, and section 4 specializes it to create a hybrid of an elitist selecto-recombinative GA with a simple hillclimber. Section 5 presents experimental results on the bit counting (onemax) problem. A simple mathematical analysis and a comparison of the model with experiment is given in section 6. Finally we suggest some extensions to this work.

## II. OUTLINE OF A THEORY OF HYBRIDIZATION

Although there are some guidelines and techniques for doing hybrids, no theory of hybridization seems to exist. Such a theory should address at least the following issues

- Costs
  1. of solving problem with a given method.
  2. of seeking better knowledge.
  3. of deciding among different methods.
- Knowledge
  1. of problem domain.
  2. of strengths and weaknesses of various methods.
- Reliability
  1. of knowledge of the problem.
  2. of knowledge of strengths and weaknesses of various methods.
- Decision making
  1. decide among different methods.
  2. decide between seeking or not for better knowledge.

Throughout the rest of the paper, our attention is focused on the decision-making problem of deciding among different methods, and all the other issues mentioned above are ignored. The next section presents a simple model to combine two methods. To keep things simple each of them is isolated into a black box so that they do not interfere directly with each other. We will be using one or the other based on which of them is more likely to help us solve the problem.

## III. DECIDE BETWEEN TWO METHODS

Assume there are two different methods to solve a particular problem. Let's call them Method A and Method B. Also, assume that the problem is solvable in a certain number of time steps by repeatedly applying either one of the methods. At each time step a choice must be made between one or the other. What is the best strategy to use? If at each time step we knew which of the methods was going to be most useful to solve the problem, we would always choose the better one. But since it is unknown beforehand which of the methods is the better one, we are left with a decision-making problem similar to the two-armed bandit problem. Moreover, we are facing a bandit that can change

its mind because which method is the better one is likely to change through time. The next paragraph illustrates the concept of probability matching and its connection to the decision-making process involving two mutually exclusive alternatives.

Suppose a two-armed bandit like the one described in [3]. On average, one of the arms is better than the other but we don't know which of the two is the better one. In that case, some experimentation is needed to figure out which of the arms is likely to be the better one. After that, the arm that was better during the experimental phase would always be chosen. Now let's imagine that we don't know anything about the bandit at all. Maybe the bandit switches its behavior once in a while. We could imagine that for a certain period of time the left arm was the better one and suddenly after some time, the bandit's right arm started paying more. If we do the experimental phase while the left arm is paying more, we will be misled and will not be able to detect the switch to the right arm. If instead, we keep experimenting all the time and allocate our trials in proportion to the payoff given by each of the arms, we could learn the bandit's behavior and be able to adjust if suddenly the right arm started giving more payoff. This later behavior is called probability matching [4] and it can be shown that it is a useful one in a non-stationary environment.

Going back to the model, we can think of the bandit's arms as corresponding to our two methods. At a given time step, one of them should be picked and we will do so using a probability matching approach. The whole idea here is to do the experimentation continuously and make the best use of the information that we have at a given time. When method A is picked we will get some reward, a measure of how well method A helped us to solve the problem. Similarly, when method B is chosen we will be rewarded by a measure of how well method B performed. In addition, we would like the later rewards to have more weight than the rewards that were obtained a long time ago. Doing so will allow us to detect a switch much faster. A simple way to achieve this is with some sort of weighted geometric average. The simple equation

$$W_{t+1} = W_t (1 - c) + c R_t.$$

where  $W_t$  denotes the weighted reward at time  $t$ ,  $R_t$  is the reward at time  $t$ , and  $c$  is a constant in  $[0..1]$ , produces such an effect. The constant  $c$  is a control parameter that tells us how far back the past rewards are remembered.  $c = 0$  means full memory (remember everything and the present reward has no effect),  $c = 1$  means no memory (just use the last reward and forget about the past). Having a measure for the payoff, we are now able to use the probability matching strategy.

This mechanism has similarities with work done on adapting operator probabilities [5], [6]. All these schemes adapt the operator probabilities based on their recent contributions to the algorithm's performance.

After presenting our simple model, we proceed to the

next section and specialize the bandit by putting a hillclimber on the left arm and a selecto-recombinative GA on the right arm.

#### IV. A SIMPLE HYBRID GA

Now method A is a hillclimber and method B is a selecto-recombinative GA. Let's look at the mechanics of each of them alone. The hillclimber picks one individual from the population, flips a randomly chosen bit, and keeps the better of the two individuals. The elitist GA picks two individuals from the population, applies uniform crossover, and keeps the two best individuals among the two parents and their two offspring. This is the same GA as the one described by Thierens and Goldberg [7]. The two methods are combined in a generational model with a fixed population size. Individuals are selected randomly and without replacement from the population. During the run, both methods will compete for trials and their probability of getting selected will adapt based on their performance.

Every time a GA step is performed, it generates two new individuals. To make things fair, whenever the HC is chosen, it is applied to two distinct individuals.

After looking at each of them separately, let's see how the ideas presented in the previous section can be applied. For both methods, the reward is measured by the difference between the fitness of the best individual after the method is applied, and the fitness of the best individual before the method is applied. In other words, there is a reward when there is an innovation, an individual that is better than what we had before. The reward measure is fair because both methods only spend one function evaluation per individual.

The updating of the weighted rewards is incremental. Every time a GA step is done on a pair of individuals, we update the weighted reward of the GA and consequently the probabilities of picking the GA and the HC on the next time step. Similarly, the same thing happens when the HC is chosen.

#### V. EXPERIMENTAL RESULTS

The hybrid algorithm was tested on the bit counting function (onemax). In this problem, the fitness value of an individual is the number of 1's in the given chromosome. The first experiments use population sizes according to Goldberg, Deb and Clark [8] and the hybrid picks the GA almost all the time. This happens because on average the expected reward of the GA is greater than the expected reward of the HC. In order to give the HC a chance, we run again our simulation with a small population size. This gives the GA a hard time and we expect that at some point the population will be almost converged to a non-optimal solution and the GA will not give much improvement. At this time we expect to observe a switch to the HC. That is exactly what happened (figure 1). Let's comment on the case where the control parameter  $c = 0.1$  (bottom graph of figure 1). The hybrid starts off using the GA roughly around 85% of the time and then it reaches a point where the population has almost converged and because of that,

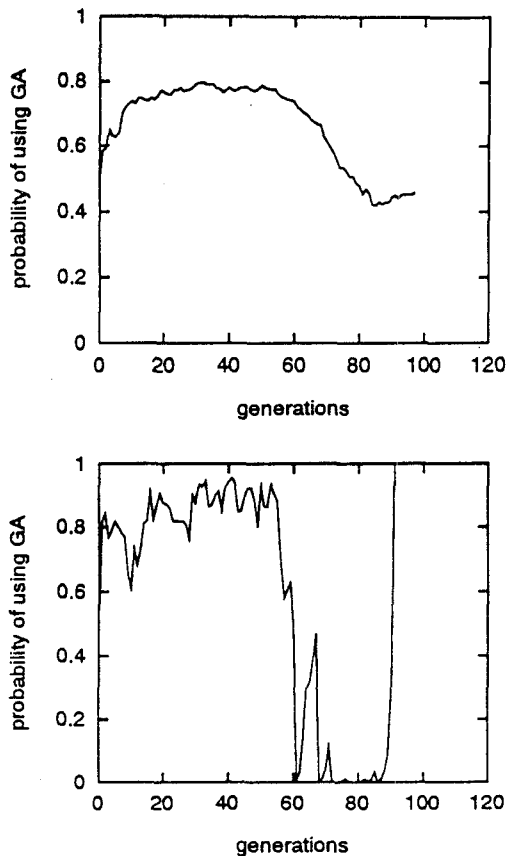


Fig. 1. Typical runs of the hybrid GA on a 200-bit problem using a population size of 30. The runs finish when all the individuals reach the optimum. On the top graph, the control parameter  $c = 0.01$ , on the bottom  $c = 0.1$ . Note that for smaller values of  $c$  the system takes longer to adapt.

crossover is not giving much improvement. The system detects it and switches to the HC. Note that at the end of the run it switches back to the GA, probably it is distributing the last correct alleles to the other individuals in the population.

On another experiment, the adaptive mechanism was switched off and the simulation was done with a 100% of GA steps (a pure GA) and then with a 100% of HC steps (a pure HC). The GA alone could never find the optimum due to the small population size. The HC alone took on average 663 generations to get an optimum individual — the combination (hybrid) performed better than the pure GA or the pure HC.

On a last set of experiments, the HC was turned off and the elitist GA was run with mutation applied after crossover. After that, the offspring are evaluated and the two best are chosen to the next generation. Again, only one function evaluation was spent per individual. Mühlenbein [9] and Bäck [10] showed that when optimizing a unimodal binary function, a mutation rate of  $1/l$  performs best (assuming a fixed mutation rate throughout the run). Their result was obtained using an asexual evolutionary algorithm. It was not extended for evolutionary algorithms with crossover. Nevertheless, if the population size is small,

a mutation rate of  $1/l$  (here, and through the rest of the paper,  $l$  denotes always the chromosome length) is still a very well tuned parameter when optimizing an easy function such as the onemax. Experiments with other mutation rates were performed and the results are summarized in table I, along with the different experiments performed with the hybrid. The results are averaged over 20 runs.

algorithm	mean	std. dev.
hybrid with $c=0.001$	93	18.7
hybrid with $c=0.01$	83	14.4
hybrid with $c=0.05$	87	20.4
hybrid with $c=0.1$	93	17.1
hybrid with $c=0.2$	88	20.9
hybrid with $c=0.5$	89	26.5
elitist GA with $\text{mut}=1/10l$	155	74.8
elitist GA with $\text{mut}=1/l$	69	9.6
elitist GA with $\text{mut}=3/l$	109	13.2
elitist GA with $\text{mut}=4/l$	162	30.1
pure HC	663	45.4
pure GA	never	—

TABLE I

Number of generations to get an optimum.  $\text{popsize} = 30$ ,  $l = 200$ .

As expected, the GA with a mutation rate of  $1/l$  performed better than the hybrid. The extra number of function evaluations performed by the hybrid is a cost of exploration, a price that had to be paid in order to learn the “optimal” mutation rate. As mentioned in section 2, the decision-making facet was isolated and we ignored all the other issues of hybridization, including the use of knowledge. The fact that a mutation rate of  $1/l$  performs well on this problem is valuable knowledge that the hybrid didn’t use. The other runs illustrate the case where the mutation rates are poorly chosen ( $1/10l$ ,  $3/l$ ,  $4/l$ ), a situation that might occur when there is no knowledge about what a good mutation rate might be. In those cases the hybrid performed better. Summarizing, the hybrid works quite well under various settings of the control parameter  $c$ , but never beats a fine tune parameter setting. This was also observed by Davis [5]. In his paper, he concluded: “... using an adaptive mechanism leads to performance slightly inferior to that of carefully-derived interpolated parameter settings ...”

## VI. MATHEMATICAL ANALYSIS

This section presents a simple mathematical analysis to try to predict what is the expected reward of each of the methods when applied alone on the onemax problem. From the empirical results we suspect that the GA gives better reward at the beginning and that there will be a point where the HC starts giving more reward.

### A. The hillclimber alone

Let  $f_t$  be the average fitness of the population at generation  $t$ . At the next generation the fitness increase will be given by the probability of flipping a 0 to a 1

$$f_{t+1} = f_t + \frac{l - f_t}{l},$$

Solving this equation and setting  $f_0$  to  $l/2$ , the expected average fitness at the initial generation, we obtain an equation that gives us the average fitness of the population at generation  $t$

$$f_t = l - \frac{l}{2} \left(1 - \frac{1}{l}\right)^t. \quad (1)$$

If we only do HC steps on the onemax problem, the fitness is a random variable with a binomial distribution and it can be approximated to the normal distribution  $N(\mu, \sigma^2)$ , where  $\mu$  and  $\sigma^2$  are the mean and variance, respectively. In our case  $\mu = l p_t$  and  $\sigma^2 = l p_t(1 - p_t)$ , where  $p_t$  is the proportion of correct alleles at generation  $t$ . Let  $X$  and  $X'$  be random variables that give the fitness distribution at generation  $t$  and  $t + 1$  respectively. Then the expected reward from generation  $t$  to  $t + 1$  is given by  $\mu_{x'_{2:2}} - \mu_{x_{2:2}}$ , where  $\mu_{x_{2:2}}$  is the mean of the second order statistic of a sample of size 2 of the random variable  $X$  [11]. Using a result from that paper we have

$$\mu_{x_{2:2}} = \mu_x + \sigma_x \frac{1}{\sqrt{\pi}}. \quad (2)$$

From equation 1, we can obtain  $p_t = f_t/l$ . Combining this with equation 2 we obtain the expected reward  $R(t)$  when going from generation  $t$  to generation  $t + 1$

$$\begin{aligned} R(t) &= \mu_{x'_{2:2}} - \mu_{x_{2:2}} \\ &= \left(\mu_{x'} + \sigma_{x'} \frac{1}{\sqrt{\pi}}\right) - \left(\mu_x + \sigma_x \frac{1}{\sqrt{\pi}}\right) \\ &= \frac{-\sqrt{(2-k)kl} + \sqrt{\frac{k(l-1)(k+2l-kl)}{l}} + k\sqrt{\pi}}{2\sqrt{\pi}}, \end{aligned} \quad (3)$$

where  $k = (1 - 1/l)^t$ . Experimental results match the theoretical ones (Figure 2).

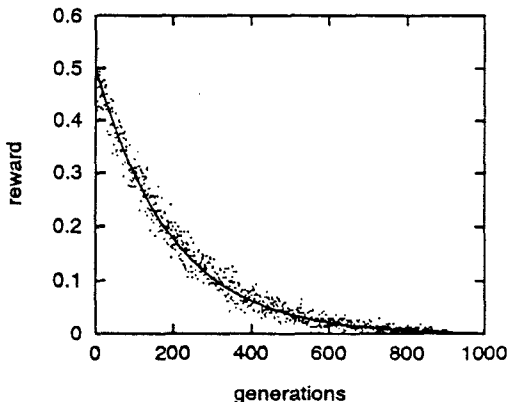


Fig. 2. Comparison of theory (solid line) and experiment (dots) using the hillclimber alone. Results are averaged for 20 runs.  $popsize = 30$ ,  $l = 200$ .

### B. The GA alone

Thierens and Goldberg [12] showed that on the onemax problem, the elitist GA behaves very much like the simple GA using binary tournament selection. More recently,

Miller and Goldberg [11], and Bäck [13] used order statistics and extended the previous model for tournament selection to account for different tournament sizes. Their model showed that for a tournament size  $s$ , the fitness improvement between generations is given by

$$\begin{aligned} \Delta\mu_{F,t} &= \mu_{F,t+1} - \mu_{F,t} \\ &= \sigma_{F,t} \mu_{s:s} \end{aligned}$$

where  $\mu_{F,t}$  and  $\sigma_{F,t}$  are the mean and standard deviation of the population fitness at generation  $t$ , and  $\mu_{s:s}$  is the maximal order statistic of size  $s$  of the standard normal distribution  $N(0, 1)$ .

Based on their results, we are able to compute the expected reward when using the GA alone. The reward computation is given by the difference between the best of parents and offspring and the best of the parents. The expected fitness of the best of parents and offspring is given by the mean of the fourth order statistic of a sample of size 4 and the expected fitness of the best of the parents is given by the mean of the second order statistic of a sample of size 2. Using this, we can derive the expected reward  $R(t)$  when going from generation  $t$  to  $t + 1$

$$\begin{aligned} R(t) &= \sigma_{F,t} (\mu_{4:4} - \mu_{2:2}) \\ &= \sqrt{l p(t) (1 - p(t))} (\mu_{4:4} - \mu_{2:2}) \end{aligned} \quad (4)$$

On equation 4,  $p(t)$  is the the proportion of correct alleles obtained by Thierens and Goldberg [12]

$$p(t) = 0.5 \left(1 + \sin\left(\frac{t}{\sqrt{\pi l}}\right)\right).$$

$\mu_{4:4}$  and  $\mu_{2:2}$  are the expected values of the maximal order statistics of sizes 4 and 2 for the standard normal distribution ( $\mu_{4:4} = 1.0294$ ,  $\mu_{2:2} = 0.5642$ )

Figure 3 shows a comparison of theory and experiment. The small discrepancy between the two is mainly because

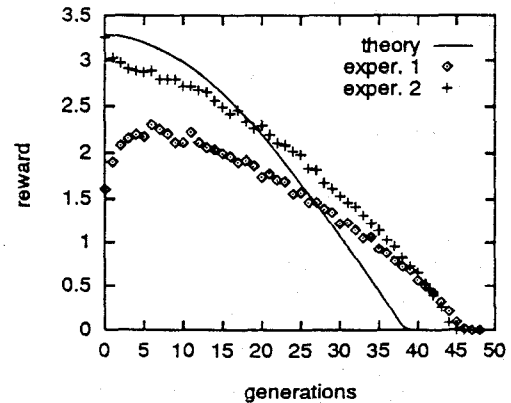


Fig. 3. Comparison of theory and experiment using the GA alone. The difference between experiments 1 and 2 is that on the second one, the individuals are shuffled after crossover. Results are averaged for 20 runs.  $popsize = 400$ ,  $l = 200$ .

the model assumes that parents and offspring are statisti-

cally independent. This is not exactly true because after crossover, the resulting offspring are highly correlated with the parents. On another experiment, we shuffle the individuals after performing crossover and only after that, the reward is computed for every pair of individuals. This way, the four individuals that are needed to compute the reward value are more statistically independent of each other. By doing so, we get a much better fit between the model and experiment. The small difference that still remains is now due to the build-up of covariances between the alleles as explained in [12].

The model is an example of what has been called elsewhere [14] of a "little model". It is not exact but it is qualitatively good, and in a practical sense, the small difference does not really alter what we can extract from it — the expected reward of the GA will be greater at the beginning of the run and at some point when the population has almost converged, the expected reward of the HC will be greater (compare figure 2 with figure 3). Note that in the case of the HC, the model is very accurate, and doing the simulation with a bigger population size would only make a closer match between the theoretical prediction and the experiment.

## VII. EXTENSIONS

We started with the onemax problem because it is easy to analyze and for a first choice it is wise to keep things as simple as possible. Also, the main reason why we chose the one-bit mutation hillclimber is because it is easy to analyze. However, the model presented here is very general and many possibilities exist for further exploration. We just need to be careful in order to make a fair comparison between the two methods. Now the system needs to be tested on a range of test functions to see how it behaves on more difficult problems.

Looking back, the hybrid presented here is still a blind searcher, it is a combination of two blind searchers. On a real application, we would probably want to incorporate previous knowledge about the problem, and by doing so, we will certainly see the real advantages of using a hybrid. Further research in this and in other facets of hybridization is needed and is going under way at the Illinois Genetic Algorithms Laboratory.

## VIII. CONCLUSIONS

The goal of this work was to get some insight into the decision-making problem on a hybrid GA. The objective was accomplished but we are aware that this is only a small part of the whole hybridization issue. There is no doubt that hybridization is important, and that any efficient real world application of genetic algorithms is most likely to be some sort of hybrid. We need to have some idea off how to do this hybridization in an efficient way. This paper started to scratch a little bit this issue using a combination of careful experimentation and a simple model based on the sound theoretical grounds of probability matching.

## ACKNOWLEDGMENTS

The first author was supported by "Junta Nacional de Investigação Científica e Tecnológica (JNICT, Portugal)" through scholarship PRAXIS BD4020/94 and by "EDP - Electricidade de Portugal". This effort was also sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant numbers F4960-94-1-0103 and F49620-95-1-0338. The first author also acknowledges the helpful discussions with Brad L. Miller.

## REFERENCES

- [1] D. J. Powell, S. S. Tong, and M. M. Skolnick, "EnGENEous domain independent, machine learning for design optimization," in *Proceedings of the Third International Conference on Genetic Algorithms*, J. D. Schaffer, Ed., San Mateo, CA, 1989, pp. 151-159, Morgan Kaufmann.
- [2] L. Davis, Ed., *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, 1991.
- [3] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, MI, 1975.
- [4] D. E. Goldberg, "Probability matching, the magnitude of reinforcement, and classifier system bidding," *Machine Learning*, vol. 5, no. 4, pp. 407-425, 1990, (Also TCGA Report No. 88002).
- [5] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, J. D. Schaffer, Ed., San Mateo, CA, 1989, pp. 61-69, Morgan Kaufmann.
- [6] B. A. Julstrom, "What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, L. Eschelman, Ed., San Francisco, CA, 1995, pp. 81-87, Morgan Kaufmann.
- [7] D. Thierens and D. E. Goldberg, "Elitist recombination: An integrated selection recombination GA," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, 1994, pp. 508-512.
- [8] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex Systems*, vol. 6, pp. 333-362, 1992.
- [9] H. Mühlenbein, "How genetic algorithms really work: I. Mutation and Hillclimbing," in *Parallel Problem Solving from Nature- PPSN II*, R. Männer and B. Manderick, Eds., Amsterdam, The Netherlands, 1992, pp. 15-25.
- [10] T. Bäck, "Optimal mutation rates in genetic search," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest, Ed., San Mateo, CA, 1993, pp. 2-8, Morgan Kaufmann.
- [11] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, no. 3, pp. 193-212, 1995.
- [12] D. Thierens and D. Goldberg, "Convergence models of genetic algorithm selection schemes," in *Parallel Problem Solving from Nature- PPSN III*, Y. Davidor, Hans-Paul Schwefel, and R. Männer, Eds., Berlin, 1994, pp. 119-129, Springer-Verlag.
- [13] T. Bäck, "Generalized convergence models for tournament- and  $(\mu, \lambda)$ -selection," *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 2-8, 1995.
- [14] D. E. Goldberg, "Toward a mechanics of conceptual machines," *Proceedings of the Eighteenth Southeastern Conference on Theoretical and Applied Mechanics*, pp. 1-9, 1996.