Missouri Estimation of Distribution Algorithms Laboratory

Initial-Population Bias in the Univariate Estimation of Distribution Algorithm

Martin Pelikan and Kumara Sastry

MEDAL Report No. 2009001

January 2009

Abstract

This paper analyzes the effects of an initial-population bias on the performance of the univariate marginal distribution algorithm (UMDA). The analysis considers two test problems: (1) onemax and (2) noisy onemax. Theoretical models are provided and verified with experiments. Intuitively, biasing the initial population toward the global optimum should improve performance of UMDA, whereas biasing the initial population away from the global optimum should have the opposite effect. Both theoretical and experimental results confirm this intuition. Effects of mutation and sampling are also analyzed and the performance of UMDA is compared to that of the mutation-based hill climbing. While for deterministic onemax the hill climbing is shown to deal with the initial bias very well, for noisy onemax performance of the hill climbing is poor regardless of the bias.

Keywords

Estimation of distribution algorithms, univariate marginal distribution algorithm, initial population bias, onemax, noisy onemax, genetic algorithm.

Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Department of Mathematics and Computer Science
University of Missouri–St. Louis
One University Blvd., St. Louis, MO 63121
E-mail: medal@cs.umsl.edu
WWW: http://medal.cs.umsl.edu/

Initial-Population Bias in the Univariate Estimation of Distribution Algorithm

Martin Pelikan
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Math and Computer Science, 320 CCB
University of Missouri at St. Louis
One University Blvd., St. Louis, MO 63121
pelikan@cs.umsl.edu

Kumara Sastry
Intel Corp.
Computational Lithography Group
Hillsboro, OR 97124

Abstract

This paper analyzes the effects of an initial-population bias on the performance of the univariate marginal distribution algorithm (UMDA). The analysis considers two test problems: (1) onemax and (2) noisy onemax. Theoretical models are provided and verified with experiments. Intuitively, biasing the initial population toward the global optimum should improve performance of UMDA, whereas biasing the initial population away from the global optimum should have the opposite effect. Both theoretical and experimental results confirm this intuition. Effects of mutation and sampling are also analyzed and the performance of UMDA is compared to that of the mutation-based hill climbing. While for deterministic onemax the hill climbing is shown to deal with the initial bias very well, for noisy onemax performance of the hill climbing is poor regardless of the bias.

Keywords: Estimation of distribution algorithms, univariate marginal distribution algorithm, initial population bias, onemax, noisy onemax, genetic algorithm, hill climbing.

1 Introduction

Since there is usually no *a priori* knowledge about the location of the global optimum and other high-quality solutions, evolutionary algorithms typically generate the initial population of candidate solutions according to the uniform distribution over all potential solutions. However, the distribution of the initial population may also be biased. The bias may come from the practitioner who can use problem-specific knowledge about the location of high-quality solutions or information obtained from previous runs on a similar problem in order to bias the initial population towards those promising areas. Furthermore, the bias can be imposed via hybridization with various local search techniques, which bias the population towards local optima. Although the initial-population bias is commonplace, the effects of biasing the initial population on performance of evolutionary algorithms have not been studied thoroughly.

The purpose of this paper is to analyze the effects of biasing the initial population in estimation of distribution algorithms. To simplify the analysis, we consider the univariate marginal distribution

```
Univariate marginal distribution algorithm (UMDA)
  t := 0;
  generate initial population P(0);
  while (not done) {
    select population of promising solutions S(t);
    learn the probability vector pv(t) for S(t);
    sample pv(t) to generate offspring O(t);
    incorporate O(t) into P(t);
    t := t+1;
  };
```

Figure 1: Pseudocode of the univariate marginal distribution algorithm (UMDA).

algorithm (UMDA), which is one of the simplest estimation of distribution algorithms. As the test problem, we consider onemax and noisy onemax. The paper provides a principled approach to biasing the initial population, theoretical models estimating the effects of such a bias, and a series experiments validating these models.

The paper starts with a description of the univariate marginal distribution algorithm in section 2. Section 3 describes the approach used to bias the initial population of candidate solutions. Section 4 describes a theoretical model for deterministic onemax and validates the model with empirical results. Section 5 describes a theoretical model for noisy onemax and verifies the model with experiments. Section 6 discusses the effects of bit-flip mutation and the sampling procedure used on UMDA performance; the section also compares the results obtained with UMDA to those obtained with simple mutation-based hill climbing. Finally, section 7 summarizes and concludes the paper.

2 Univariate Estimation of Distribution Algorithm

The univariate marginal distribution algorithm (UMDA) (Mühlenbein & Paaß, 1996) uses a probabilistic model in the form of a probability vector (Baluja, 1994; Juels, Baluja, & Sinclair, 1993) to model and sample candidate solutions. Here we assume that candidate solutions are represented by fixed-length binary vectors, although it is straightforward to consider alphabets of higher cardinality. The probability vector considers only univariate probabilities and for each string position it stores the probability of a 1 in that position. For an n-bit string, the probability vector is thus a vector of n probabilities $p = (p_1, p_2, \ldots, p_n)$ where p_i encodes the probability of a 1 in the i-th position.

UMDA starts with a population of candidate solutions generated according to the uniform distribution over all binary strings. Promising solutions are then selected using any popular selection method of genetic algorithms. In this paper we use binary tournament selection. A probability vector is then learned from the selected solutions; the ith entry of the probability vector is set to the proportion of ones in the i-th position of the selected population. New candidate solutions are sampled from the distribution encoded by the probability vector. The new solutions are then incorporated into the original population using a replacement strategy; for example, the new solutions can fully replace the original population of strings. See Figure 1 for the pseudo-code of UMDA.

3 Biasing Initial Population on Onemax

Although evolutionary algorithms typically generate the initial population of candidate solutions according to the uniform distribution over all potential solutions, in some cases the distribution of the initial population may be biased. Ideally, the initial-population bias would be expected to move the population closer to the global optimum; nonetheless, this is rarely guaranteed and any initial-population bias may have the opposite effect.

To understand and predict the effects of the initial-population bias, one of the key questions is whether the bias moves the population closer to or further from the global optimum. Of course, other properties of the initial population are important as well, such as the initial supply of important building blocks (Goldberg, Sastry, & Latoza, 2001; Goldberg, 2002) and the diversity of the population. Increasing the initial supply of important building blocks moves the population closer to the optimum and decreasing the initial supply of important building blocks moves the population further away from the optimum. Generally, any initial-population bias is expected to reduce population diversity.

As the starting point in the analysis of the initial-population bias, we will use the simple onemax fitness function:

$$onemax(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} X_i, \tag{1}$$

where X_i denotes the i-th bit of the input binary vector of n bits. Onemax is a simple unimodal linear function with the optimum in the string of all 1s. The more 1s in the solution, the higher the fitness. A 1 in any position can thus be considered as a building block and the global solution is the string combining all these building blocks. With a sufficiently large and diverse population of binary strings, UMDA is expected to increase the proportion of 1s in any single position until the entire population converges to the global optimum of onemax.

To bias the initial population, we simply generate each bit with a specified probability p_{init} or $p(0)$. For the uniform distribution, $p_{init} = 0.5$. As p_{init} increases, the population is expected to contain more 1s on average, leading to higher-quality solutions. On the other hand, as p_{init} decreases, the population is expected to contain fewer 1s, leading to lower-quality solutions. Essentially, varying p_{init} allows us to control the initial supply of important building blocks and the expected distance of the population and its members from the global optimum.

Of course, other approaches to biasing the initial population may be proposed; nonetheless, we believe that the aforementioned approach provides a great starting point for analyzing the effects of the initial-population bias on performance of estimation of distribution algorithms on additively decomposable problems.

4 Mathematical Model for Onemax

This section discusses the population-sizing and time-to-convergence models which can be used to predict performance of UMDA on onemax with varying levels of bias of the initial population. These models are then validated by experiments.

4.1 Population Size

The model presented in this section is based on the gambler's ruin population sizing theory (Harik, Cantú-Paz, Goldberg, & Miller, 1997; Harik, Cantú-Paz, Goldberg, & Miller, 1999). The gambler's

ruin model assumes a steady-state genetic algorithm with binary tournament selection. While UMDA is not a steady state algorithm, it has been shown that the gambler's ruin model is applicable also to standard, generational genetic algorithms (Harik, Cantú-Paz, Goldberg, & Miller, 1997). The fitness function is assumed to be a separable function with identical (or at least similar) subproblems. This assumption is clearly satisfied for onemax because the contribution of each bit (string position) is independent of other bits and each bit contributes to the overall fitness in the same way; each string position can thus be taken as an independent subproblem with the optimum 1. It is also assumed that an aggressive crossover operator is used that strongly decorrelates different string positions. This assumption is also clearly satisfied since UMDA considers each string position independently and the strings generated by UMDA are not expected to contain any correlations between different string positions.

The gambler's ruin model characterizes the subproblems by two quantities as proposed by Goldberg et al. (Goldberg & Rudnick, 1991; Goldberg et al., 1992): the signal, d, and the variance, σ^2_{bb}. The signal denotes the difference between the fitness contribution of the optimal assignment of bits in one subproblem and that of the second best assignment. The variance denotes the variance of the fitness values for this subproblem. For onemax, the signal is $d = 1$ (the contribution of a 1 is 1 and the contribution of a 0 is 0) and the variance is $\sigma^2_{bb} = 0.25$ under the assumption that the numbers of 0s and 1s in this string position are equal. The maximum possible variance of the fitness contribution of a single string position in onemax is 0.25.

Let us consider one round of binary tournament selection and denote by p the probability that a 1 in a string position wins the tournament; the probability that a 0 wins is denoted by $q = 1 - p$. Since the average fitness of solutions with a 1 in this string position is expected to be greater than the average fitness of solutions with a 0 in this position, we would expect that $p > q$. Assuming that p and q do not change with time, the gambler's ruin model can be used to compute the probability that a 1 overtakes the entire population (Feller, 1970; Harik et al., 1997):

$$p_{bb} = \frac{1 - \left(\frac{q}{p}\right)^{x_0}}{1 - \left(\frac{q}{p}\right)^{N}} \tag{2}$$

where N is the population size, $p_{init} \in [0, 1]$ is the initial proportion of 1s in each position in the initial population, and $x_0 = p_{init} \times N$ is the expected initial number of 1s in the population.

Assuming the uniform distribution over all binary strings and the onemax fitness function, the probability of a 1 winning over a 0 in one string position can be estimated as (Goldberg et al., 1992; Harik et al., 1997)

$$p = \Phi\left(\frac{d}{\sqrt{n'/2}}\right), \tag{3}$$

where $n' = n - 1$ and Φ is the cumulative distribution function for a normal distribution with zero mean and unit standard distribution.

While p and q do change with the distribution of candidate solutions, the probability of a 1 winning a tournament increases as the fitness variance decreases. Thus, since the uniform distribution of candidate solutions maximizes the overall fitness variance, the assumption that p remains constant according to equation 3 is a conservative one and can be expected to provide us with an upper bound on the population size.

Substituting $p_{bb} = 1 - \alpha$ into equation 2, where α is the probability of error (probability that

the bit converges to 0 instead of 1), results in

$$1-\alpha = \frac{1-\left(\frac{q}{p}\right)^{p_{init}N}}{1-\left(\frac{q}{p}\right)^{N}}. \tag{4}$$

After making several approximations and incorporating equation 3 into the last equation, we can isolate N as (Harik et al., 1997)

$$N = -\frac{1}{4p_{init}} \ln \alpha\sqrt{\pi n'}. \tag{5}$$

Since for moderate to large n, $n' = n - 1 \approx n$, we get

$$N = -\frac{1}{4p_{init}} \ln \alpha\sqrt{\pi n}. \tag{6}$$

While this result is based on several strong assumptions, the experiments will confirm a close match between this model and the actual performance of UMDA (see Section 4.4).

Note that for onemax with $p_{init} < 0.5$, the fitness variance does not monotonically decrease with time. Instead, the variance increases to its maximum at $n/4$ and then decreases to 0. The decision making between 1s and 0s is most difficult when the variance reaches its maximum. Nonetheless, since our model assumes that the fitness variance is always at its maximum, this does not affect the final result, which provides a conservative estimate.

4.2 Number of Generations

Let us assume that the probability of any bit being 1 at time t is the same for all string positions and that binary tournament selection is used. Furthermore, let us assume that crossover combines bits effectively and that the underlying problem is the onemax of n bits. Finally, let us assume that the initial probability of any bit being 1 is p_{init}. Then, the probability of a particular bit being 1 at time t is given by (Thierens & Goldberg, 1994)

$$p(t) = 0.5\left(1+\sin\left(\frac{t}{\sqrt{\pi n}} + \arcsin(2p_{init}-1)\right)\right). \tag{7}$$

Setting probability $p(t) = 1$ then allows us to compute the upper bound on the expected time to convergence:

$$1 = 0.5\left(1+\sin\left(\frac{G}{\sqrt{\pi n}} + \arcsin(2p_{init}-1)\right)\right) \tag{8}$$

$$1 = \sin\left(\frac{G}{\sqrt{\pi n}} + \arcsin(2p_{init}-1)\right) \tag{9}$$

$$\frac{\pi}{2} = \frac{G}{\sqrt{\pi n}} + \arcsin(2p_{init}-1) \tag{10}$$

$$G = \left(\frac{\pi}{2} - \arcsin(2p_{init}-1)\right)\sqrt{\pi n} \tag{11}$$

Experiments presented in section 4.4 confirm that the final model provides us with a conservative upper bound on the number of generations for UMDA on onemax across a range of values of p_{init}.

4.3 Predicting Effects of Initial-Population Bias

Clearly, both N and G depend on p_{init}. Intuitively, starting with fewer 1s in the initial population should lead to larger values of N and G, and thus decreasing p_{init} should result in increasing N and G. To approximate how these two terms change with respect to p_{init}, we can use the population-sizing and time-to-convergence models from the previous two subsections.

Let us denote the population size and the time to convergence for a specific value of p_{init} by $N_{p_{init}}$ and $G_{p_{init}}$. To provide a practical measure of the effects of p_{init} on the computational complexity of UMDA on onemax, let us consider the case $p_{init} = 0.5$ as the basis case and compute the factor by which the population size and the number of generations change for other values of p_{init}. The factor for the population size will be denoted by η_N, the factor by which the number of generations changes will be denoted by η_G. From equation 6, we get

$$\eta_N = \frac{N_{p_{init}}}{N_{0.5}} \tag{12}$$

$$\eta_N = \frac{1}{2p_{init}} \tag{13}$$

For the number of generations, we get

$$\eta_G = \frac{G_{p_{init}}}{G_{0.5}} \tag{14}$$

$$\eta_G = \frac{\frac{\pi}{2} - \arcsin(2p_{init} - 1)}{\frac{\pi}{2}} = 1 - \frac{2\arcsin(2p_{init} - 1)}{\pi} \tag{15}$$

With respect to the overall number of evaluations given by the product of the population size and the number of generations, the factor by which the number of evaluations changes compared to the case with $p_{init} = 0.5$ is given by

$$\eta_E = \frac{N_{p_{init}} G_{p_{init}}}{N_{0.5} G_{0.5}} = \eta_N \times \eta_G. \tag{16}$$

Substituting equations 13 and 15 into equation 16 gives us

$$\eta_E = \frac{1}{2p_{init}} \left(1 - \frac{2\arcsin(2p_{init} - 1)}{\pi} \right). \tag{17}$$

The three slowdown factors and the corresponding speedup factors are shown in Figure 2. These results confirm the initial intuition that the population size, the number of generations, and the number of evaluations decrease with increasing p_{init} and they all increase with decreasing p_{init}. Furthermore, the results show that the factor by which these quantities change does *not* depend on problem size. Nonetheless, the growth of the population size, the number of generations and the number of evaluations for relatively small values of p_{init} is relatively fast. For example, for $p_{init} = 0.05$, the number of evaluations increases by a factor of more than 17.

4.4 Empirical Validation of Theory

To validate the theoretical models presented earlier in this section, we ran a number of experiments with UMDA. The experiments considered the onemax of $n = 100$ to $n = 500$ bits, but the largest

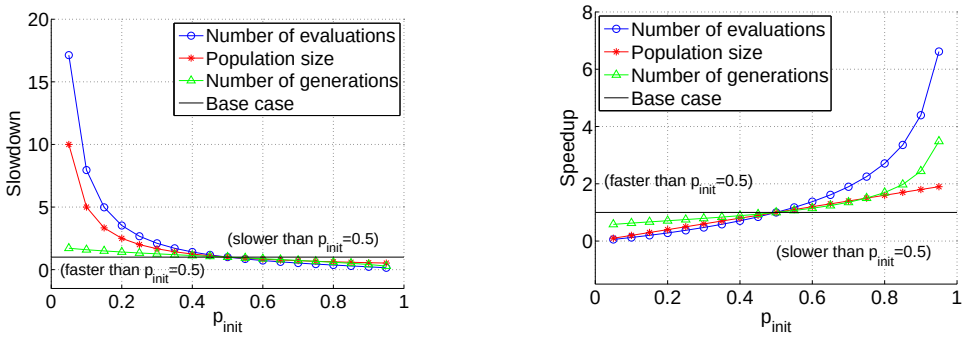


Figure 2: The factor by which the population size, the number of generations and the number of evaluations should change with varying p_{init} compared to the base case with $p_{init} = 0.5$. The three factors are based on the population-sizing and time-to-convergence models. The results are shown as speedup and slowdown curves.

problem of $n = 500$ is used in most presented results. In all experiments, binary tournament selection *without* replacement is used. The sampling ensures that the generated population of strings corresponds to the the used probability vector as closely as possible in order to minimize sampling errors; this is done by first generating the expected number of 1s for each position and then randomly shuffling the bits in each position across the population of new strings (Branke, Lode, & Shapiro, 2007). In each generation, a population of the same size as the original population is generated and the old population is fully replaced by the newly generated strings. There is no elitism or niching. To alleviate the effects of stochastic noise on the initial-population bias, the initial population is generated similarly as the new solutions in each iteration; first, the expected number of 1s is put in every string position and the bits are then shuffled randomly.

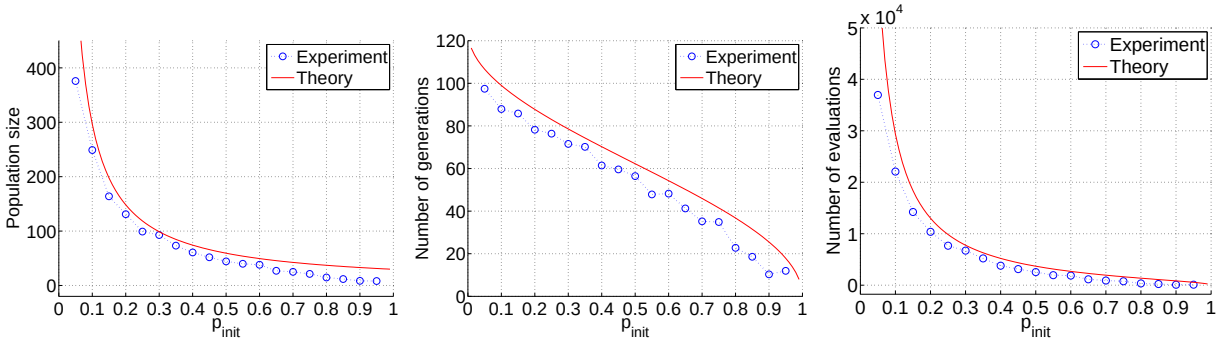
The bisection method (Sastry, 2001; Pelikan, 2005) was used to find an adequate population size for each setting, which ensured that the algorithm finds a good-enough solution in each of the 10 independent runs. In each run, UMDA was required to find a solution with at least 95% of string positions containing a 1 (at least 95% bits of the global optimum were correctly discovered). To provide more accurate estimates of the actual UMDA performance, the bisection was repeated 10 times for each setting, yielding 100 successful runs for each configuration.

The results for UMDA are shown in Figure 3. The results indicate a close match between the theoretical model and experimental results. Analogical match was observed for other problem sizes.

5 Mathematical Model for Noisy Onemax

Genetic algorithms are known to perform well even when the fitness cannot be determined exactly due to external noise. The noise may come from an inherently noisy problem domain or a noisy approximation of a computationally expensive fitness function. External noise can also be used to model certain types of problem difficulty. This section extends the analysis presented thus far to onemax with external Gaussian noise. This will provide us with the basic understanding of the interaction between the initial-population bias and external noise.

The section starts with the description of the noisy onemax function. Next, the section presents the population-sizing and time-to-convergence models for noisy onemax. Finally, the models are verified with experiments and the results are discussed.



(a) Population size. (b) Number of generations. (c) Number of evaluations.

Figure 3: Effects of initial-population bias on UMDA performance on onemax of $n = 500$ bits without external noise.

5.1 Noisy Onemax

Noisy fitness functions can be modeled as (Goldberg, Deb, & Clark, 1992)

$$fitness' = fitness + noise, \tag{18}$$

where $fitness$ is the true fitness of the candidate solution and $noise$ is a random variable corresponding to the external noise. External noise can be characterized by two basic quantities: bias and variance (Sastry, 2001). Unbiased noise only adds variance to the population without affecting the expected fitness of any candidate solution. While increasing the fitness variance makes decision making between competing partial solutions or building blocks more difficult, the expected outcome of the computation should remain the same, assuming that the population size is increased to average out the added variance. On the other hand, biased noise may change the expected fitness of any candidate solution. This will lead to both a more difficult decision making as well as the potential for changing the overall outcome, since the best solution of the noisy fitness may not represent the best solution with no external noise.

In models of noisy fitness functions, external noise is typically distributed according to a zero-mean Gaussian distribution, which is also the considered here. The noisy onemax is thus defined as

$$onemax_{noisy}(X_1, X_2, \ldots, X_n) = onemax(X_1, X_2, \ldots, X_n) + N(0, \sigma_N^2), \tag{19}$$

where $N(0, \sigma_N^2)$ is a random variable distributed according to the Gaussian distribution with mean 0 and variance σ_N^2.

Intuitively, the effects of external noise depend on how its variance compares the overall variance of the fitness function. If the external-noise variance is very small compared to the fitness variance, the effects of external noise can be expected to be negligible; on the other hand, if the noise variance is larger than the fitness variance, the effects of external noise can be expected to be substantial. That is why the strength (variance) of external noise is often measured relatively to the fitness variance:

$$\sigma_N^2 = \beta \sigma_F^2, \tag{20}$$

where σ_F^2 is the fitness variance and $\beta \geq 0$ is a constant factor. In this paper, we consider four levels of external noise: $\beta = \{0.5, 1, 1.5, 2\}$. Note that for the considered range of β, the effects of noise on the fitness function are substantial. For the smallest $\beta = 0.5$, the variance of the external noise is half the overall fitness variance, and for the largest $\beta = 2$, the variance of the added noise is twice the fitness variance.

5.2 Population Size

The gambler's ruin population sizing model can be extended to accommodate external noise in a straightforward manner (Harik et al., 1997). Compared to the model introduced in section 4.1, the only component that changes is the estimation of the probability of a 1 winning a tournament over a 0 (Harik, Cantú-Paz, Goldberg, & Miller, 1997):

$$p = \Phi\left(\frac{d}{\sqrt{2(0.25n' + \sigma_N^2)}}\right), \tag{21}$$

Without presenting details of the derivation, the population-sizing equation boils down to (Harik et al., 1997; Harik et al., 1999)

$$N = -\frac{1}{2p_{init}} \ln \alpha \frac{\sqrt{\pi(n'\sigma_{bb}^2 + \sigma_N^2)}}{d}. \tag{22}$$

Since for moderate to large n, $n' = n - 1 \approx n$, and since for onemax $d = 1$, we get

$$N = -\frac{1}{2p_{init}} \ln \alpha \sqrt{\pi(n\sigma_{bb}^2 + \sigma_N^2)}. \tag{23}$$

Rewriting the last equation using $\sigma_N^2 = \beta\sigma_F^2 = \beta n\sigma_{bb}^2$, we get

$$N = -\frac{1}{2p_{init}} \ln \alpha \sqrt{\pi n\sigma_{bb}^2(1 + \beta)}. \tag{24}$$

Substituting $\sigma_{bb}^2 = 0.25$ into the last equation yields

$$N = -\frac{1}{4p_{init}} \ln \alpha \sqrt{\pi n(1 + \beta)}. \tag{25}$$

Thus, compared to the deterministic case, the population size is expected to increase by about a factor of $(1 + \beta)$ where $\beta = \sigma_F^2/\sigma_N^2$, regardless of p_{init}.

5.3 Number of Generations

The time-to-convergence model for deterministic functions proposed by Thierens and Goldberg (1994) was extended to functions with external noise by Miller and Goldberg (Miller & Goldberg, 1995; Miller & Goldberg, 1996). The resulting model was further simplified by Sastry and Goldberg (Sastry, 2001; Goldberg, 2002) as follows:

$$G = \frac{\pi}{2I}\sqrt{n}\sqrt{1 + \frac{\sigma_N^2}{\sigma_F^2}}, \tag{26}$$

where I denotes the selection intensity. Since for binary tournament selection, $I = 1/\sqrt{\pi}$, we get

$$G = \frac{\pi}{2}\sqrt{\pi n}\sqrt{1 + \frac{\sigma_N^2}{\sigma_F^2}}. \tag{27}$$

Therefore, the factor by which the number of generations changes compared to the deterministic case is $\sqrt{1+\beta}$ where $\beta = \sigma_F^2/\sigma_N^2$. However, in both these models it is assumed that $p_{init} = 0.5$.

The convergence model for noisy fitness functions is relatively complex and it is not straightforward to derive a closed-form expression that would provide the number of generations until convergence in terms of p_{init}. That is why we started by running experiments across a range of values of n and β. The results were then used to provide an empirical model applicable to noisy problems with moderate amount of noise.

Based on experimental results, the model for $p_{init} = 0.5$ was adjusted by the same factor as that derived for deterministic onemax (see equation 15). This led to the following empirical model for noisy onemax

$$G = \frac{\pi}{2}\sqrt{\pi n}\sqrt{1+\frac{\sigma_N^2}{\sigma_F^2}}\left(1 - \frac{2\arcsin(2p_{init}-1)}{\pi}\right), \tag{28}$$

or, in terms of β,

$$G = \frac{\pi}{2}\sqrt{\pi n}\sqrt{1+\beta}\left(1 - \frac{2\arcsin(2p_{init}-1)}{\pi}\right), \tag{29}$$

5.4 Predicting Effects of Initial-Population Bias

Based on the results presented in previous two subsections, we can compute how the population size and the number of generations change with p_{init} compared to the base, unbiased case with $p_{init} = 0.5$. These factors are identical to those derived for the deterministic problem and they thus do not appear to be influenced by external noise.

For the population size, we get

$$\eta_N = \frac{1}{2p_{init}} \tag{30}$$

For the number of generations, we get

$$\eta_G = 1 - \frac{2\arcsin(2p_{init}-1)}{\pi} \tag{31}$$

Finally, for the number of evaluations, we get

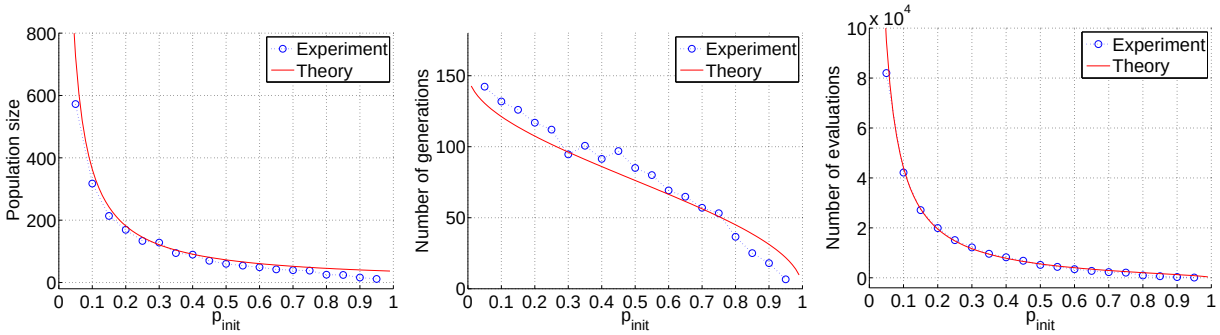
$$\eta_E = \frac{1}{2p_{init}}\left(1 - \frac{2\arcsin(2p_{init}-1)}{\pi}\right). \tag{32}$$

5.5 Empirical Validation of Theory

Here we validate the models presented in the previous few subsections on noisy onemax with $\beta = \{0.5, 1.0, 1.5, 2.0\}$. The experimental methodology was the same as that for the deterministic onemax and we thus omit the details here (see section 4.4).

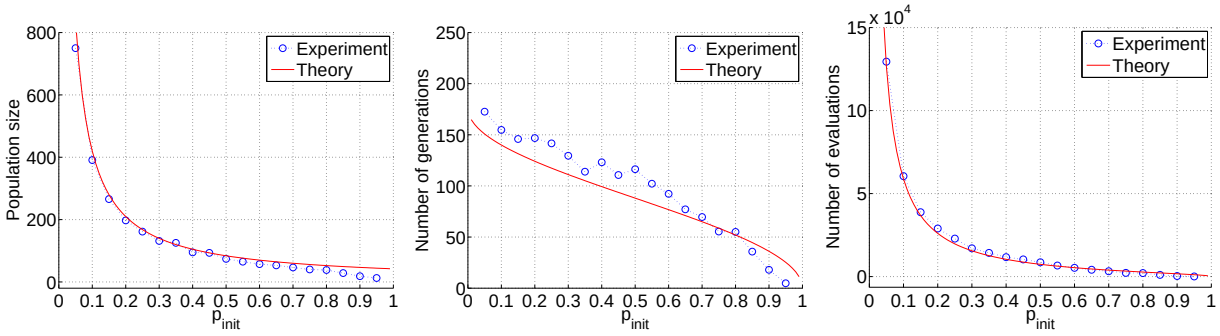
Figure 4 shows the results for $\sigma_N^2 = 0.5\sigma_F^2$ (that is, $\beta = 0.5$). Figure 5 shows the results for $\sigma_N^2 = \sigma_F^2$ (that is, $\beta = 1$). Figure 6 shows the results for $\sigma_N^2 = 1.5\sigma_F^2$ (that is, $\beta = 1.5$). Figure 7 shows the results for $\sigma_N^2 = 2\sigma_F^2$ (that is, $\beta = 2$).

In all cases, there is a close match between the theory and the empirical results, although for the number of generations the differences between the model and the experiment grow slightly as the strength of external noise increases. Nonetheless, even for $\beta = 2$, the model provides an accurate prediction of the influence of p_{init} on UMDA performance on noisy onemax.



(a) Population size. (b) Number of generations. (c) Number of evaluations.

Figure 4: Effects of initial-population bias on UMDA performance on onemax of $n = 500$ bits, $\sigma_N^2 = 0.5\sigma_F^2 = 0.125n$.



(a) Population size. (b) Number of generations. (c) Number of evaluations.

Figure 5: Effects of initial-population bias on UMDA performance on onemax of $n = 500$ bits, $\sigma_N^2 = \sigma_F^2 = 0.25n$.

Figure 8 visualizes the effects of external noise on the number of evaluations. Comparing the overall number of evaluations confirms the intuition that the number of evaluations grows with the strength of external noise. Specifically, from the models presented earlier, the number of evaluations grows linearly with the ratio of the variance of external noise and the fitness variance, β.

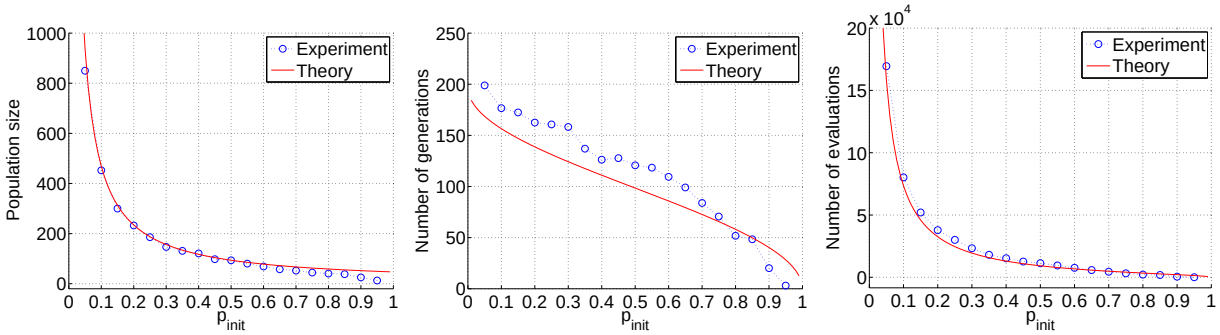
6 Effects of mutation and the sampling procedure

Thus far we examined the effects of the initial population bias on the performance of standard UMDA. This section answers three additional questions related to this topic:

(1) How does *bit-flip mutation* affect performance of UMDA with initial population bias?

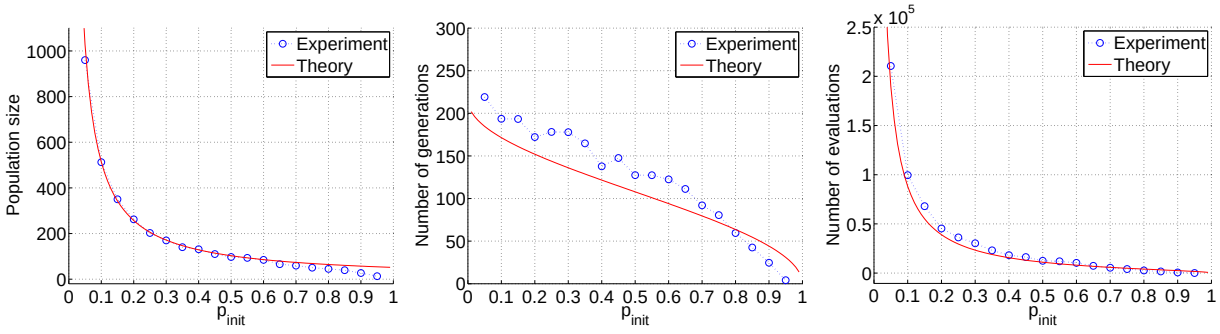
(2) How does *the sampling procedure* affect performance of UMDA with initial population bias?

(3) How is performance of *mutation-based hill climbing* affected by initial bias and noise compared to UMDA?



(a) Population size. (b) Number of generations. (c) Number of evaluations.

Figure 6: Effects of initial-population bias on UMDA performance on onemax of $n = 500$ bits, $\sigma_N^2 = 1.5\sigma_F^2 = 0.375n$.



(a) Population size. (b) Number of generations. (c) Number of evaluations.

Figure 7: Effects of initial-population bias on UMDA performance on onemax of $n = 500$ bits, $\sigma_N^2 = 2\sigma_F^2 = 0.5n$.

6.1 Incorporating Mutation into UMDA

For decades there has been an ongoing debate on the importance of crossover and mutation in genetic algorithms. UMDA is a pure recombination-based search, as the learning and sampling the probability vector corresponds to repeated application of crossover (Mühlenbein & Paaß, 1996). How would mutation affect the results? Intuitively, adding mutation should help UMDA deal with the initial population bias more efficiently; nonetheless, as will be discussed shortly, the results indicate that this is not the case and that adding mutation does not really help regardless of the initial population bias.

With bit-flip mutation, for each new solution, each bit is flipped with a given probability p_{flip}. To incorporate bit-flip mutation into UMDA, we can change the probability vector used for sampling as follows:

$$p_i = p_i \left(1 - \frac{p_{flip}}{2}\right) + (1 - p_i)\frac{p_{flip}}{2} \tag{33}$$

Sampling the modified vector corresponds to the combination of standard sampling and bit-flip mutation.

The results obtained with UMDA with mutation and their comparison to those obtained with standard UMDA are shown in Figure 9. The results show that for a broad range of values of p_{init}, UMDA with mutation is less sensitive to the value of p_{init}. Nonetheless, UMDA with mutation

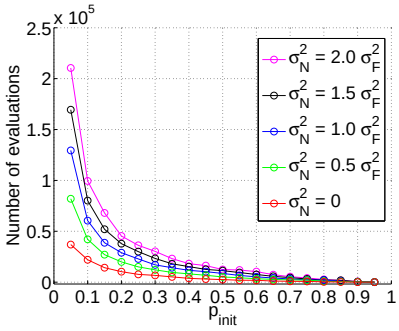
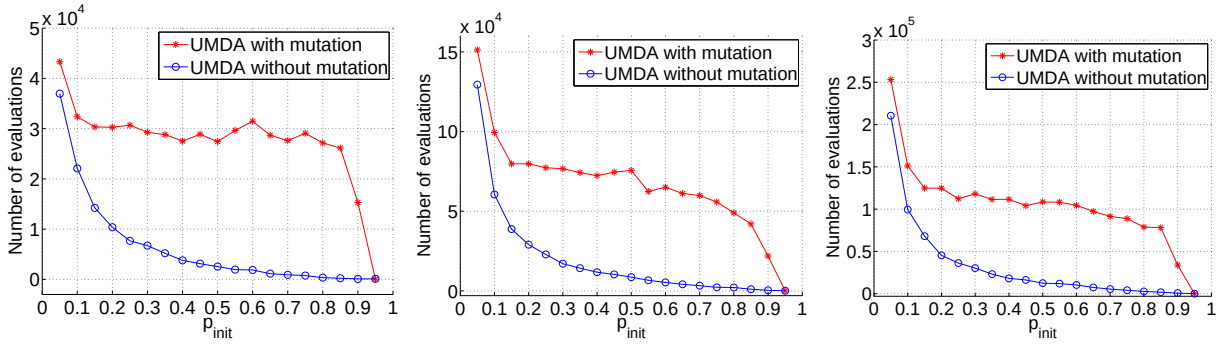


Figure 8: Effects of initial-population bias and noise on UMDA performance on onemax of $n = 500$ bits.



(a) No external noise. (b) Noise with $\sigma_N^2 = \sigma_F^2$. (c) Noise with $\sigma_N^2 = 2\sigma_F^2$.

Figure 9: Effects of mutation on UMDA performance for onemax of $n = 500$ bits.

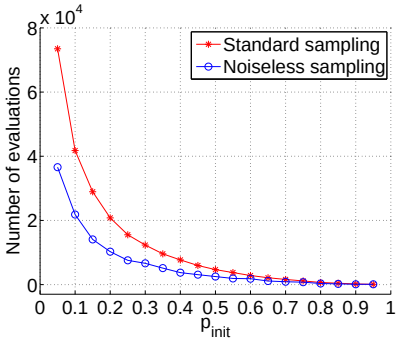
performs consistently worse than standard UMDA. Similar conclusions hold for the deterministic and noisy onemax.

6.2 Effects of Sampling

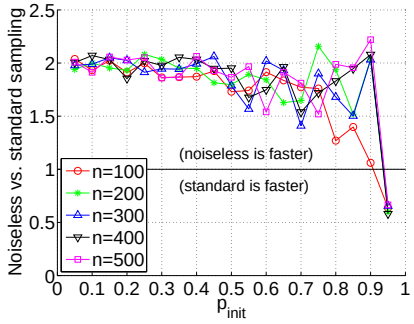
As discussed earlier, to eliminate sampling errors, we used the noiseless sampling procedure which ensures that the proportion of 1s in each position is exactly equal to the proportion stored in the sampled probability vector (Branke, Lode, & Shapiro, 2007). It was argued elsewhere (Branke, Lode, & Shapiro, 2007) that this sampling approach improves UMDA performance on onemax.

Here we investigate how the effects of noiseless sampling change with p_{init} and the variance of external noise. More specifically, we compare noiseless sampling and standard sampling, which generates each bit in each solution independently. The expected results of the two procedures are the same; nonetheless, the noiseless sampling procedure ensures that the generated sample corresponds exactly to the proportions in the probability vector whereas the standard procedure is expected to deviate from the probability vector slightly due to stochastic sampling errors.

The results of the comparison of noiseless sampling and standard sampling on deterministic onemax are shown in Figure 10. Analogical results for noisy onemax of 500 bits with $\sigma_N^2 = \sigma_F^2$ and $\sigma_N^2 = 2\sigma_F^2$ are shown in figures 11 and 12. Clearly, the use of noiseless sampling proves to be beneficial for almost all values of p_{init} regardless of external noise and problem size. The only exception are the large values of p_{init} where the stochastic errors of standard sampling appear to help. The speedup obtained with noiseless sampling is about 2 in most cases. Overall, the results

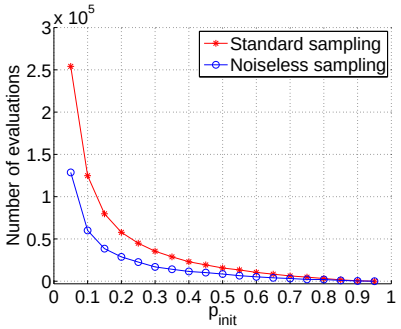


(a) Number of evaluations.

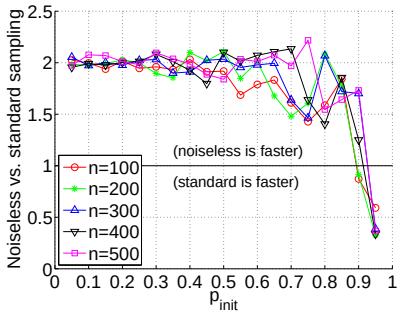


(b) Speedup of noiseless over standard sampling.

Figure 10: Effects of sampling on UMDA performance without external noise.



(a) Number of evaluations.



(b) Speedup of noiseless over standard sampling.

Figure 11: Effects of sampling on UMDA performance with external noise $\sigma_N^2 = \sigma_F^2$.

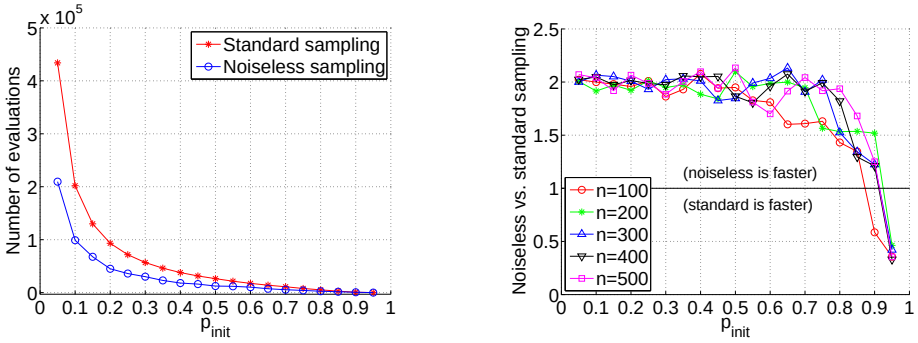
confirm that UMDA benefits from the use of noiseless sampling.

6.3 Mutation-Based Hill Climbing

Another way to incorporate mutation into search is to use the mutation-based hill climbing (HC). HC starts with a random binary string X. In each iteration, bit-flip mutation is applied to the current solution X to create a new binary candidate solution X'. After evaluating X', the two solutions are compared and if X' is better than X, X is replaced with X'. Since the probability of flipping a bit with the mutation operator is typically small, the solution generated in each iteration is likely to differ from the current solution in only one or a few bits. The mutation-based hill climbing is thus expected to explore the search space by walking to a local optimum, changing only few bits at a time.

6.3.1 Performance of Hill Climbing on Deterministic Onemax

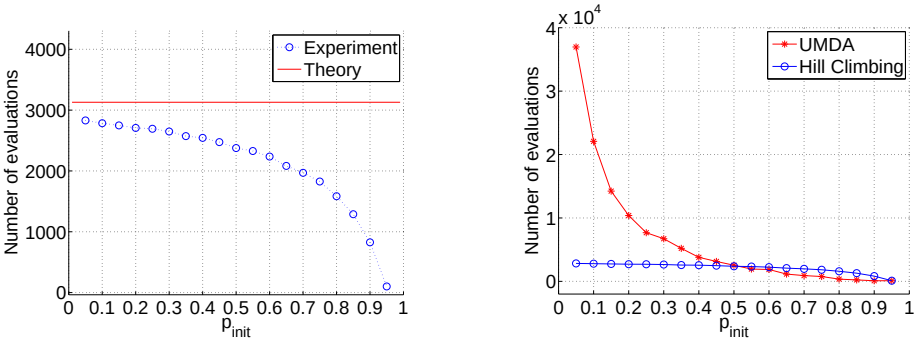
The probability of flipping a bit with mutation was set as $p_{flip} = 1/n$ as suggested by theory of Mühlenbein (1992), who showed that this setting maximizes HC performance on onemax. 100 independent runs of HC were performed for onemax of size from $n = 100$ to $n = 500$. Similarly as for UMDA, each run was terminated when a solution was found that contained at least 95% of



(a) Number of evaluations.

(b) Speedup of noiseless over standard sampling.

Figure 12: Effects of sampling on UMDA performance with external noise $\sigma_N^2 = 2\sigma_F^2$.



(a) Hill climbing on 500-bit onemax.

(b) Comparison of UMDA and HC.

Figure 13: Performance of HC on a 500-bit deterministic onemax and its comparison to UMDA. Theoretical model of Mühlenbein (Mühlenbein, 1992) is used to provide an upper bound on the time to convergence (on the left).

1s. The total number of evaluations until termination was then recorded and averaged over all 100 runs. To analyze the influence of the initial-population bias on HC performance, the initial string is generated to contain $p_{init}n$ ones.

It is well known that HC works very well on onemax and similar unimodal problems (Mühlenbein, 1992). This is confirmed with the results shown in Figure 13. HC is clearly much less sensitive to the value of p_{init} than UMDA. This fact is not surprising because it was shown (Mühlenbein, 1992) that in HC majority of computational resources are typically spent in making the last few steps toward the optimum. Furthermore, the results clearly indicate that on onemax HC significantly outperforms UMDA except for relatively large values of p_{init}. The smaller p_{init}, the greater the differences between HC and UMDA.

6.3.2 Performance of Hill Climbing on Noisy Onemax

After several initial experiments, it became clear that while on deterministic onemax HC significantly outperforms UMDA, on noisy onemax the situation changes rapidly. This result is not a big surprise, because it was shown that recombination-based search can deal with external noise much more effectively than search based on local operators (Sastry & Goldberg, 2004); nonetheless, the

β	n	p_{init}	HC evaluations	UMDA evaluations
0.5	10	0.1	4,449	1,210
0.5	25	0.1	2,125,373	1,886
0.5	10	0.5	11,096	66
0.5	25	0.5	8,248,140	169
1.0	5	0.1	215	574
1.0	15	0.1	5,691,725	1,210
1.0	5	0.5	64	20
1.0	15	0.5	15,738,168	64

Table 1: Performance of HC on noisy onemax.

differences were much larger than we expected.

HC performance on noisy onemax was so poor that even for the simplest case with $\beta = 0.5$, HC was not able to solve problems of size 30 or more in practical time. To illustrate the poor performance of HC on noisy onemax, we summarize some of the results in Table 1. The results show that HC performance is dramatically worse than that of UMDA and that while UMDA can easily solve noisy onemax instances of hundreds of bits regardless of β, HC cannot solve instances of even 30 bits or more. Additional experiments showed that HC cannot deal well with even much weaker external noise (we omit these results).

7 Summary and Conclusions

Most applications of evolutionary algorithms incorporate efficiency enhancement techniques, such as hybridization with some form of local search. Many of these techniques may result in a bias of the initial population. Yet, the effects of the population bias have not received much attention. This paper presented an attempt to gain a better understanding of the effects of the initial-population bias on performance of a simple estimation of distribution algorithm.

More specifically, this paper studied the effects of the initial-population bias on performance of the univariate marginal distribution algorithm (UMDA). As test problems, onemax and noisy onemax were used. Theoretical models were provided and verified with experiments. The results showed that when the initial population is strongly biased away from the global optimum, UMDA performance gets substantially worse than with the uniformly distributed initial population. On the other hand, the population bias is beneficial if it moves the population toward the global optimum. The multiplicative factor quantifying performance improvement was shown to be independent of problem size. Mutation was empirically shown to alleviate the effects of the initial-population bias although the performance of UMDA with mutation is considerably worse than that of standard UMDA. It was empirically confirmed that using noiseless sampling which alleviates stochastic errors in sampling improves UMDA performance on onemax regardless of the existence or strength of external noise and regardless of the initial-population bias. While on a deterministic onemax mutation-based hill climbing was shown to be much more efficient and resistant to the population bias than UMDA, on a noisy onemax the hill climbing was shown to perform extremely poorly.

There are several promising avenues for extending this research; here we mention a few important ones. First of all, specific efficiency enhancement techniques should be studied with respect to the initial-population bias. Most importantly, two approaches should be considered: (1) Hybridization and (2) online and offline use of prior problem-specific knowledge. Second, the analysis should be extended to other types of evolutionary algorithms, especially to estimation of distribution al-

gorithms based on multivariate models such as the hierarchical Bayesian optimization algorithm. Third, techniques should be proposed to alleviate the negative effects of the initial-population bias whenever possible. Fourth, for specific problem types and specific bias techniques, the distribution of the initial bias could be approximated and the models presented in this paper could then be used to quantify expected effects.

Acknowledgments

This project was sponsored by the National Science Foundation under CAREER grant ECS-0547013, by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant FA9550-06-1-0096, and by the University of Missouri in St. Louis through the High Performance Computing Collaboratory sponsored by Information Technology Services, and the Research Award and Research Board programs.

The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the Air Force Office of Scientific Research, or the U.S. Government.

References

Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning* (Tech. Rep. No. CMU-CS-94-163). Pittsburgh, PA: Carnegie Mellon University.

Branke, J., Lode, C., & Shapiro, J. L. (2007). Addressing sampling errors and diversity loss in umda. pp. 508–515.

Feller, W. (1970). *An introduction to probability theory and its applications*. New York, NY: Wiley.

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms*, Volume 7 of *Genetic Algorithms and Evolutionary Computation*. Kluwer Academic Publishers.

Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, *6*, 333–362.

Goldberg, D. E., & Rudnick, M. (1991). Genetic algorithms and the variance of fitness. *Complex Systems*, *5*(3), 265–278. Also IlliGAL Report No. 91001.

Goldberg, D. E., Sastry, K., & Latoza, T. (2001). On the supply of building blocks. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 336–342. Also IlliGAL Report No. 2001015.

Harik, G., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, *7*(3), 231–253.

Harik, G. R., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Proceedings of the International Conference on Evolutionary Computation (ICEC-97)*, 7–12. Also IlliGAL Report No. 96004.

Juels, A., Baluja, S., & Sinclair, A. (1993). The equilibrium genetic algorithm and the role of crossover.

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, *9*(3), 193–212.

Miller, B. L., & Goldberg, D. E. (1996). Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, *4*(2), 113–131.

Mühlenbein, H. (1992). How genetic algorithms really work: I.Mutation and Hillclimbing. In Männer, R., & Manderick, B. (Eds.), *Parallel Problem Solving from Nature* (pp. 15–25). Amsterdam, Netherlands: Elsevier Science.

Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature*, 178–187.

Pelikan, M. (2005). *Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms*. Springer.

Sastry, K. (2001). *Evaluation-relaxation schemes for genetic and evolutionary algorithms*. Master's thesis, University of Illinois at Urbana-Champaign, Department of General Engineering, Urbana, IL. Also IlliGAL Report No. 2002004.

Sastry, K., & Goldberg, D. E. (2004). Let's get ready to rumble: Crossover versus mutation head to head.

Thierens, D., & Goldberg, D. (1994). Convergence models of genetic algorithm selection schemes. *Parallel Problem Solving from Nature*, 116–121.