Addressing Sampling Errors and Diversity Loss in UMDA

Jürgen Branke
Institute AIFB
University of Karlsruhe
76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

Clemens Lode
Institute AIFB
University of Karlsruhe
Karlsruhe, Germany
clemens@lode.de

Jonathan L. Shapiro
School of Computer Science
University of Manchester
Manchester, M13 9PL UK
Jonathan.Shapiro@manchester.ac.uk

ABSTRACT

Estimation of distribution algorithms replace the typical crossover and mutation operators by constructing a probabilistic model and generating offspring according to this model. In previous studies, it has been shown that this generally leads to diversity loss due to sampling errors. In this paper, for the case of the simple Univariate Marginal Distribution Algorithm (UMDA), we propose and test several methods for counteracting diversity loss. The diversity loss can come in two phases: sampling from the probability model (offspring generation) and selection. We show that it is possible to completely remove the sampling error during offspring generation. Furthermore, we examine several plausible model construction variants which counteract diversity loss during selection and demonstrate that these update rules work better than the standard update on a variety of simple test problems.

Categories and Subject Descriptors

I2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

UMDA, variance loss, sampling error

1. INTRODUCTION

Estimation of Distribution Algorithms (EDAs) are search algorithms which use probability models instead of populations to search for solutions of a problem. A population is generated by sampling from the current probability model and selection is applied to this population. The selected population is then used to generate the probability model for the next generation, using methods developed by the statistical machine learning community to build probability models from data. A range of EDAs have been proposed and developed, both for continuous and discrete search spaces, and a number of successful applications have been reported. For reviews, see the book [6] or the review article [9].

One known problem of EDAs is that they may suffer from "premature convergence". As diversity is lost from the probability model, it will not be restored. This was first shown in [4, 10] for PBIL, and has been shown to hold generally for EDAs [12]. The effect is analogous to genetic drift[1] for ordinary evolutionary algorithms, and it is most apparent when operating on a flat fitness landscape, as the algorithm quickly converges to a single solution.

The variance loss happens at two stages of the EDA: when the population is generated by sampling from the probability distribution, and when selection is applied to this population to generate the probability model for the next generation. In this paper, we propose a number of ways to counteract this variance loss for the example of the Univariate Marginal Distribution Algorithm (UMDA), which is a typical yet simple EDA. However, we believe that similar ideas can be applied also to more complex EDAs.

As we show, the variance loss in the offspring generation phase can be easily removed. Counteracting the variance loss due to selection is more difficult, and we propose and empirically compare several approaches.

The paper is structured as follows. In Section 2 we give a short introduction to UMDA and briefly recall the main results on variance loss. Section 3 deals with removing the variance loss from the offspring generation phase, while different alternatives to reduce the variance loss in the selection phase are discussed in Section 4. The different proposed variants are empirically compared in Section 5. The paper concludes with a summary and some ideas for future work.

2. UMDA

The Univariate Marginal Distribution Algorithm (UMDA) [8] is an EDA which acts on discrete variables and treats all variables as independent. For simplicity, we consider binary strings of length L. The probability model treats each component of the string independently. The parameters of the model are the probabilities that each component takes the value 1, denoted,

$$\gamma_i \equiv P(x_i = 1). \quad (1)$$

At each iteration, N strings will be sampled. The ith com-

[1]In population genetics, the term *drift* refers to the loss of genetic diversity due to finite population sampling

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

ponent of the μth string is denoted x_i^μ. Thus, the probability of generating a string $\mathbf{x}^\mu$ can be written

$$P(\mathbf{x}^\mu) = \prod_{i=1}^{L} \left[\gamma_i x_i^\mu + (1-\gamma_i)\left(1-x_i^\mu\right)\right], \tag{2}$$

and the probability of producing a population of N strings is just the product of Equation (2) over μ.

Truncation selection is typically used although any selection operator can be utilized; the selected population consists of a fraction f of the best fitnesses in the population (e.g. $f = 0.5$). The parameters of the new model are given by the frequencies of 1's at each site in the selected population.

$$\gamma_i \leftarrow \frac{1}{(fN)} \sum_{\mu \in D_s} x_i^\mu, \tag{3}$$

where D_s is the data from the selected population. This choice of parameters maximizes the likelihood of the selected data. Algorithm 1 shows the algorithm.

Algorithm 1 Simple UMDA algorithm

Set $\gamma_i \leftarrow 1/2$ for all $i = 1 \dots L$;
repeat
  Sample N strings according to Equation (2) to make a population D.
  Generate a new population D_s from D by selecting the fN fittest strings.
  for $i = 1$ to L **do**

$$\gamma_i \leftarrow \frac{1}{(fN)} \sum_{\mathbf{x}^\mu \in D_s} x_i^\mu, \tag{4}$$

  end for
until Stopping criterion met

As has been shown in [11, 12], EDAs, and UMDA in particular, suffer from a variance loss over generations. In brief, instead of performing an unbiased search, they tend to search in regions of the search space where they have searched before. To see why this is, consider what happens when the algorithm is run on a flat landscape where all the states are equally good and selection has no effect on the statistics of the population. Obviously, the expected mean of any property of the sampled population would be the mean of the probability model. However, the variance of the sampled population will be less than that of the probability model. It is very well known that the variance of a sample of size N has a expected value of size $\sigma^2(1-1/N)$ where σ^2 is the variance in the parent distribution. Most EDAs do not compensate for this; when the new probability model is produced, it attempts to model the new population and therefore has a reduced variance. When this is iterated repeatedly, the variance of the sampled population gets smaller and smaller and decays to zero. The probability model evolves to one which can only generate identical configurations.

The diversity loss in UMDA occurs in two steps:

1. When sampling from the model distribution to generate the N offspring solutions. This happens independently of the optimization problem at hand and is due to the well-known fact that a sampled population has a variance smaller than that of the distribution from which it was drawn.

2. When performing selection on the N sampled solutions. If selection is random, e.g. because there are several individuals with equivalent fitness, or because a randomized selection method is used, then this is just equivalent to further sampling. In general, some diversity loss is due to the population focusing on fitter solutions, and some is random, but it is difficult to analyze and is problem dependent. We will use model generation rules to mitigate against this loss.

For the dynamics on a flat landscape, it was shown in [12] that if the diversity at a single bit position in a population of size N is σ^2 (as measured by the variance), then after removing a fraction $(1-f)$ of the population by truncation selection, constructing a model and generating a new population of size N, the variance is reduced to $\sigma^2\left(1-\frac{1}{fN}\right)$. In other words, the total variance loss within one iteration on a flat landscape corresponds to the factor

$$\mathcal{L}_t = \left(1 - \frac{1}{fN}\right). \tag{5}$$

This is distributed on the two sources of sampling error as follows: Because we know that sampling a population of size N from a distribution reduces the variance by a factor $(1-\frac{1}{N})$, the variance loss of generating N offspring involves a variance loss of

$$\mathcal{L}_g = \left(1 - \frac{1}{N}\right) \tag{6}$$

The selection step thus is responsible for the remaining variance loss,

$$\mathcal{L}_s = \frac{fN-1}{fN-f}. \tag{7}$$

In standard EAs, the sampling error in the case of generational reproduction is usually reduced by stochastic universal sampling [1]. This idea has also been transferred to steady-state EAs in [3]. Of course, standard EAs usually have a mutation operator which explicitly changes the statistics of the current population towards a more randomized one. Many EDAs lack this, attempting to faithfully reproduce the statistics of the current population in the constructed probability model.

The importance of diversity loss in EDAs was probably first investigated by Gonzalez and collaborators [5, 4]. The authors showed that in PBIL, a independent variable EDA related to UMDA, once the probability at a particular site got sufficiently close to zero or one, the probability that the diversity would be restored at that site can be arbitrarily close to zero. This could be avoided if the rate of updating the probability model was sufficiently slow, so that the diversity loss is slower than the search rate. However, how slow this rate must be is very problem-dependent, as was shown in [10].

Several methods have been used to reduce diversity loss. Although the early authors did not explicitly mention diversity loss, early EDAs use selection mechanisms which update the probability model very slowly. For example, in an early study of an EDA called MIMIC [2], statistics about the populations visited were maintained in a matrix and this matrix

was used to build the probability model. The current population had a (roughly) 1% effect on this matrix.

For UMDA, the rate of unintended diversity loss depends on the population size, and is reduced with increasing population size. The appropriate population size is again problem dependent with an exponential (in the string length) dynamic range [11]. A wide range of EDAs, including those which learn structural relationships between variables, behave like UMDA in this regard [12].

An explicit approach to control diversity loss in EDAs was introduced by Mahnig and Mühlenbein [7]. They used Laplace's method of updating the probability model; a method which is considered further in Section 4.2 of this paper. This approach is equivalent to a Bayesian method of updating. It was shown to have an analogous effects to mutation, and was deemed "Bayesian mutation". Like mutation, it is an operator which when repeatedly applied converges to the random population.

Another approach is to impose a detailed-balance condition on the probability updates [10, 11]. This approach is parameter-free and completely removes diversity loss on a flat landscape and for some problems it is very effective. However, it makes it difficult to sample the optimum in some problems. Cross-validation using a second population to test for diversity loss due to random sampling was employed by structure modeling in [13], to some effect, but was not applied in UMDA.

3. REMOVING THE VARIANCE LOSS DUE TO SAMPLING

The variance loss due to sampling can be easily removed by ensuring that the generated population has exactly $N\gamma_i$ individuals with bit i set to $''1''$. This can be achieved by simply generating all bits for all the N individuals, and then randomly permuting the bits amongst the individuals in each bit position. Also, note that because γ_i are calculated according to Equation 4, $N \cdot \gamma_i$ is always integer, and the variance loss can be removed completely. Later on, we will propose alternative model construction rules for γ_i which do no longer have this property. In this case, we suggest to randomly round up or down proportional to the rounding error. That is, $N \cdot \gamma_i$ is rounded up to $\lceil N\gamma_i \rceil$ with probability $N\gamma_i - \lfloor N\gamma_i \rfloor$ and rounded down otherwise.

We call the proposed sampling mechanism *permutation sampling*. The goal and effect is similar to the goal and effect of stochastic universal sampling [1] used for parent selection in generational evolutionary algorithms, namely to remove sampling errors due to small sample sizes (i.e., a small population size) as much as possible.

4. REDUCING THE VARIANCE LOSS DUE TO SELECTION

As discussed in Section 2, the variance loss due to selection on a flat landscape corresponds to $\mathcal{L}_s = \frac{fN-1}{fN-f}$. In this section, we propose a couple of ways to remove the effect of this variance loss. Note that it is not possible to remove the variance loss by intelligent sampling, as we have done for the offspring generation, because we can only select whole individuals, and the gene positions are thus strongly interdependent.

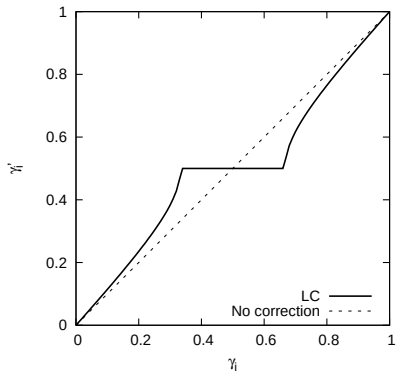

Figure 1: Effect of loss correction (LC) on γ_i, assuming $N = 10$.

4.1 Loss correction

One idea to remove the effect of the expected variance loss is to bias the model to generate a larger variance. Because we model binomial distributions, the variance of the probability model at gene position i is just $\sigma_i^2 = \gamma_i(1-\gamma_i)$. To counteract the variance loss, we would thus replace γ_i by a γ_i' such that the expected variance loss is counterbalanced

$$\gamma_i'(1-\gamma_i') = \frac{\gamma_i(1-\gamma_i)}{\mathcal{L}_s} \tag{8}$$

As in a binomial distribution the highest variance is obtained for $\gamma_i = 0.5$ and decreases towards 0 and 1, this basically corresponds to moving γ_i closer to 0.5. Equation 8 easily leads to

$$\gamma_{i,1/2}' = \frac{1 \pm \sqrt{1 - 4(1-\gamma_i)\gamma_i/\mathcal{L}_s}}{2} \tag{9}$$

For this to be a valid expression, we require $1 - 4(-\gamma_i^2 + \gamma_i)/\mathcal{L}_s \geq 0$. This is fulfilled only for $\gamma_i \leq \frac{1}{2}\left(1 - \sqrt{1 - \frac{fN-1}{fN-f}}\right)$ and $\gamma_i \geq \frac{1}{2}\left(1 + \sqrt{1 - \frac{fN-1}{fN-f}}\right)$, in which case we can use the above equation to counterbalance the expected variance loss. For γ_i within these bounds, a $\gamma_i' = 0.5$ would not be sufficient to counterbalance the variance loss, in which case we simply set $\gamma_i' = 0.5$.

Overall, the corrected probability is calculated according to

$$\gamma_i' = \begin{cases} \frac{1-\sqrt{1-4(1-\gamma_i)\gamma_i/\mathcal{L}_s}}{2} & : \quad \gamma_i \leq \frac{1}{2}\left(1 - \sqrt{1-\mathcal{L}_s}\right) \\ \frac{1+\sqrt{1-4(1-\gamma_i)\gamma_i/\mathcal{L}_s}}{2} & : \quad \gamma_i \geq \frac{1}{2}\left(1 + \sqrt{1-\mathcal{L}_s}\right) \\ 0.5 & : \quad \text{otherwise} \end{cases} \tag{10}$$

Figure 1 demonstrates how γ_i' differs from γ_i. The correction depends on the population size, N, and is smaller for larger population sizes. The strongest effect is for γ_i close to 0.5, when γ_i' is always moved back to 0.5. At the extremes, i.e. for $\gamma_i \to 0$ and $\gamma_i \to 1$, the correction vanishes. This basically leads to a stronger exploration in the early phase of the optimization, while allowing to fully converge eventually. In the following, we will denote this method as loss correction, or LC for short.

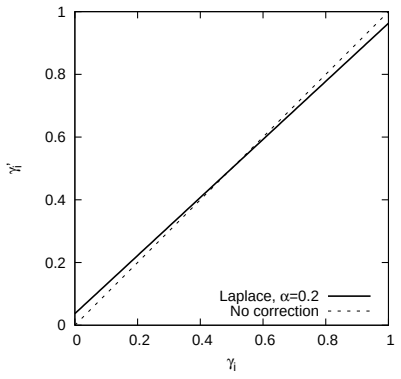

Figure 2: Effect of Laplace correction on p_i, with $N = 10$ and $\alpha = 0.2$.

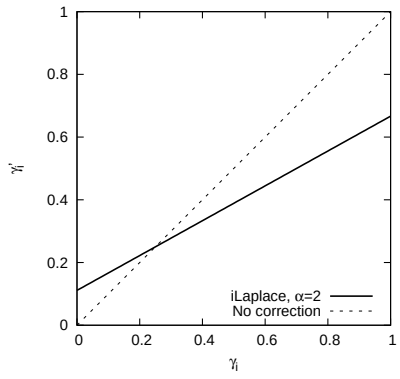

Figure 3: Effect of incremental Laplace correction on $\gamma_i(t)$, at the example of $\gamma_i(t-1) = 0.25$.

4.2 Laplace correction

Bayesian statistics usually takes into account a prior distribution when determining the probability model from observations. In the absence of any other knowledge, usually non-informative prior is used. For the binomial distributions we consider, a commonly used prior is the beta distribution, which is the conjugate prior for the binomial distribution. This would be integrated into the model construction phase by replacing Equation 4 by

$$\gamma_i = \frac{\sum_{\mathbf{x}^\mu \in D_s} x_i^\mu + \alpha}{fN + 2\alpha} \tag{11}$$

where α is a parameter which determines the strength of the influence of the prior. In fact, the beta distribution is a two-parameter distribution, but we only consider priors peaked at 1/2.

The effect on γ_i is depicted in Figure 2 for $\alpha = 0.2$. As can be seen, the effect is quite different from the loss correction described in the previous section. The probabilities are almost not influenced near $\gamma_i = 0.5$, but γ_i is prevented from converging completely to 0 or 1. This means that UMDA will never completely converge but keep exploring, similar to an EA with a minimum mutation probability. While this certainly improves the exploratory power of the algorithm, it may dramatically delay the algorithm finding the optimum, e.g. in OneMax, when the considered string is long, and it becomes very unlikely to generate an offspring consisting of all ones as long as $\gamma_i > \epsilon$. Furthermore, we know from the discussion above that the variance loss is at maximum for $\gamma_i = 0.5$, but the Laplace correction has basically no effect there.

4.3 Incremental Laplace correction

The motivation for the Laplace correction was actually to take into account a prior distribution. But except for the initial population, the population has usually been generated from a Binomial distribution with $\gamma_i \neq 0.5$. Thus, another idea would be to use beta distribution peaked at γ_i from the last iteration as the prior distribution for the current generation, instead of always assuming a prior distribution peaked at 1/2.

More formally, we set

$$\gamma_i(t) = \frac{\sum_{\mathbf{x}^\mu \in D_s} x_i^\mu + 2\alpha\gamma_i(t-1)}{fN + 2\alpha} \tag{12}$$

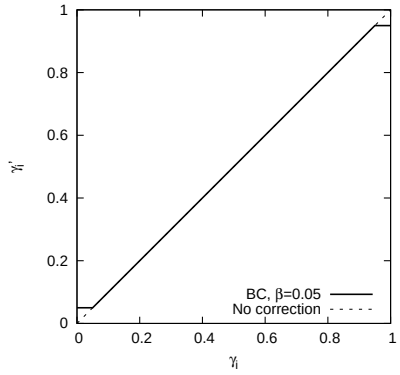

Figure 4: Effect of boundary correction on γ_i, for $\beta = 0.05$.

This can not be visualized in 2D, as it additionally depends on the previous iteration's γ_i, but Figure 3 shows the plot for $\gamma_i(t-1) = 0.25$. Basically, the incremental Laplace method, or iLaplace method for short, discourages large deviations from the previous distribution in either direction, and thus slows down convergence but does not prevent it completely as with the Laplace correction from the previous section.

As the beta distribution is a two-parameter family of functions, the constraint that the prior distribution is peaked at $\gamma_i(t-1)$ can be realized by a line in the parameter space. Equation 12 is just one way of parameterizing this line. Although any parametrization is equivalent when $\gamma_i(t-1)$ is known, $\gamma_i(t-1)$ is a fluctuating quantity. Equation 12 is the parametrization which is unchanged on a flat fitness landscape when averaged over sampling fluctuations.

4.4 Boundary correction

As has been explained above, the Laplace correction prevents UMDA from completely converging, which has a great impact on algorithm performance (good and bad, depending in the considered scenario). Because we wanted to separate the probability distribution correction effect from the non-convergence effect, we introduce here the boundary correction (BC), which does not influence the probability distribution except preventing γ_i from moving too close to either

0 or 1:

$$\gamma_i' = \begin{cases} \beta & : \quad \gamma_i < \beta \\ 1-\beta & : \quad \gamma_i > 1-\beta \\ \gamma_i & : \quad \text{otherwise} \end{cases} \tag{13}$$

.

For a visualization, see Figure 4. This method can be easily combined with the other correction methods, by simply applying it as a second stage correction.

5. EMPIRICAL EVALUATION

We start our empirical evaluation in the first subsection with an examination of the effect of removing the sampling error from the offspring generation phase. Then, in the remaining subsections, we will compare the different model update strategies on a number of simple test problems, namely

- Flat landscape: all bit strings have equal fitness values, and consequently selection is random.
- OneMax: the fitness is the number of ones in the bit string.
- LeadingOnes: the fitness is the number of leading ones in the bit string. Compared to OneMax, this leads to a random drift of the later bit positions in the early generations, because bit positions after the first zero have no influence on the fitness.
- NK landscapes allow to set the level of epistasis between bit positions. The fitness value is a sum of local fitness values f_i, where f_i is a randomly generated function based on bit position i and the $k-1$ closest neighbors. In the tests below, we use $L = 50$ and $k = 5$.

As default values and unless specified otherwise, we use a population size of $N = 20$, a fraction of individuals to select of $f = 0.5$, and problem size of $L = 100$, $L = 300$, and $L = 500$ for the flat landscape, OneMax, and LeadingOnes, respectively. Diversity is measured as average variance over all bit positions. All results reported below are averaged over 50 independent runs.

5.1 Removing the sampling error from offspring generation

The undesired diversity loss due to sampling errors is most significant on the flat landscape, because selection is purely random. Figure 5 visualizes the dramatic diversity loss at the example of a population of size $N = 20$ and a problem of size $L = 100$. According to the dashed line representing the standard UMDA, the algorithm has basically converged to a single solution after about 40 iterations. Removing the sampling error for the offspring generation step as proposed in Section 3 (solid line) can not prevent the complete loss of diversity, but at least delays it until after about 40 iterations. Nevertheless, such a delayed diversity loss can greatly improve performance on problems which provide a fitness gradient. Figure 6 shows the fitness of the best solution found so far for the OneMax problem, this time with problem size $L = 300$. Neither of the two algorithms consistently finds the optimum, but while the standard UMDA converges on average to a fitness of 241, using permutation sampling for the offspring generation pushes this up to about 272.2.

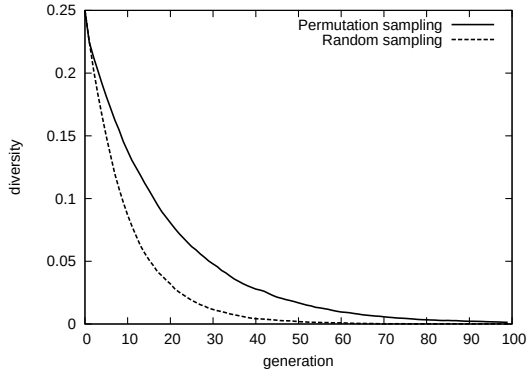

Figure 5: Diversity loss on a flat landscape, $N = 20, L = 100$.

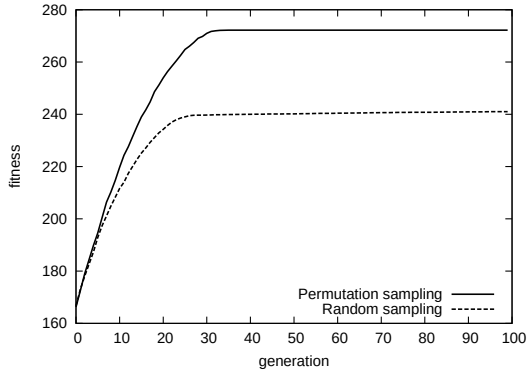

Figure 6: Convergence curves for OneMax, best solution found so far, $N = 20, L = 300$.

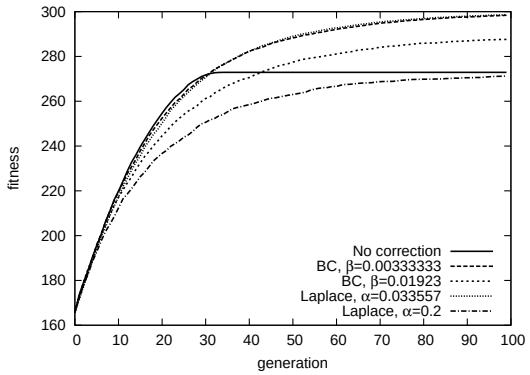

Figure 7: Comparison of BC and Laplace correction on a OneMax problem, best solution found so far, $N = 20, L = 300$.

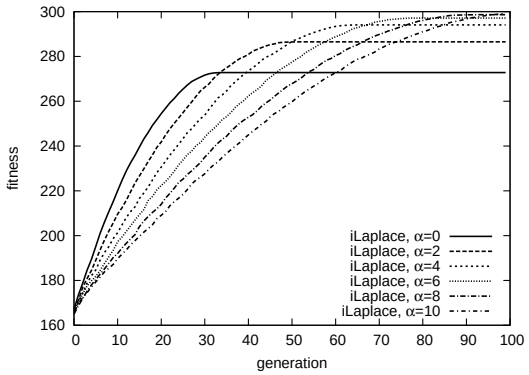

Figure 8: The effect of parameter α on the iterative Laplace correction on a OneMax problem, $N = 20, L = 300$.

Note that the removal of sampling error from the offspring generation phase does not bias the algorithm in any way but simply helps it to do what it is supposed to do. Because the variance loss depends on the population size N, the benefit from permutation sampling will be larger the smaller the population size, but it should always be beneficial. We therefore recommend using this method under all circumstances, and we use it as default in all subsequent tests to examine the different model update strategies. That is, the "No correction" case used for comparison in the following figures is already using the improved permutation sampling.

5.2 Parameter settings for update strategies

Some of our proposed model update strategies have parameters, and in this section we discuss how to set them.

The boundary correction (BC) basically corresponds to a minimal mutation probability in evolutionary algorithms. And there, for simple problems as the OneMax problem, a bit mutation rate of $1/L$ is generally recommended (with L denoting the string length of the solution). For this reason, we will set $\beta = 1/L$ in the experiments reported below unless stated otherwise. The fact that this is a good parameter setting was also confirmed in some additional empirical tests (not shown).

For Laplace correction, we assume that its main effect also lies in effectively guaranteeing a minimal mutation probability. Therefore, we set α such that the minimal mutation probability guaranteed equals $1/L$, i.e.

$$\frac{\alpha}{fN + 2\alpha} = \frac{1}{L} \tag{14}$$

or

$$\alpha = \frac{fN}{L - 2} \tag{15}$$

which is essentially also what is recommended in [7]. Figure 7 shows at the example of a OneMax problem that BC and Laplace correction with the recommended parameter setting indeed perform very similar, and much better than the standard approach. For larger parameter values, both approaches suffer, but Laplace suffers significantly more than BC.

For iLaplace correction, we have no intuitive parameter setting, and indeed the optimal parameter setting seems to depend very much on the problem at hand. Figures 8 and 9 depict the convergence of the iLaplace correction for different settings of α on the OneMax and LeadingOnes problems, respectively. While on the OneMax problem, we have a clear trade-off between convergence speed and final solution quality (a larger α leads to slower convergence but better final solutions), on the LeadingOnes problem the slower convergence is almost not noticeable, while there is a clear and dramatic increase in final solution quality. If combined with BC, iLaplace yields only a marginal improvement over pure BC, but delaying convergence significantly. Based on these observations, we set $\alpha = 2$ for iLaplace in the following, unless specified otherwise.

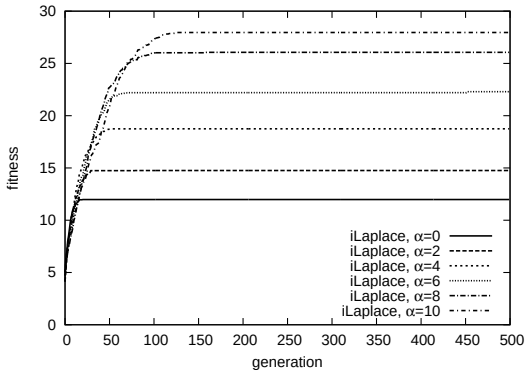

Figure 9: The effect of parameter α on the iterative Laplace correction on a LeadingOnes problem, $N = 20, L = 100$.

5.3 Comparison of update strategies

5.3.1 Flat landscape

Figure 10 compares the evolution of diversity for the different model update strategies on the flat landscape. Naturally, the BC and Laplace correction methods prevent a complete diversity loss, and we observe a somewhat higher convergence level for Laplace than for BC. LC and iLaplace both fully converge, although slower than with the standard model update strategy.

The fact that the run with LC converges shows that this

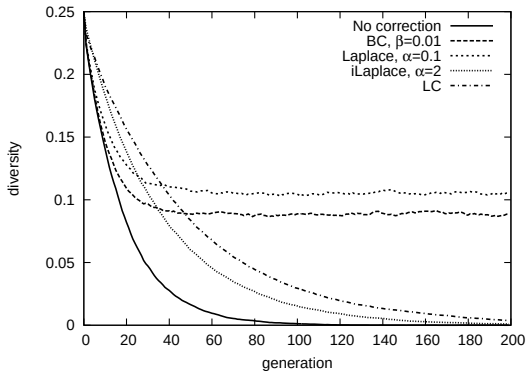

Figure 10: Comparison of the different model update strategies on the flat landscape, $N = 20, L = 100$.

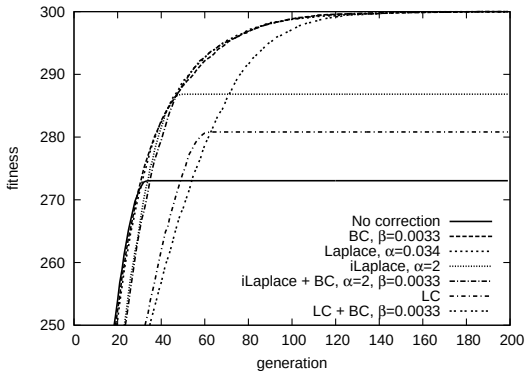

Figure 11: Comparison of the different model update strategies on the OneMax landscape, best fitness found so far, $N = 20, L = 300$.

method is not able to fully compensate the variance loss in the selection step. One reason is that it can not increase variance beyond $\gamma_i = 0.5$, another one is that it can only compensate for the *expected* loss, but not recover if the loss happened to be larger due to random fluctuations.

Interestingly, the diversity level is much higher than the minimum value of $\gamma_i = \beta$ would suggest. Considering $p(1-p) = 0.1$ yields $p \approx 0.11$, i.e. a lot more bits are mutated than we anticipated. This indicates that the results about optimal mutation ratesmight not readily carry over to BC settings in UMDA.

5.3.2 OneMax

On the OneMax problem, all proposed methods clearly outperform the standard model update in terms of fitness obtained, see Figure 11. All methods which provide a minimal diversity threshold get very close to the optimum. LC and LC+BC converge significantly slower than the other methods. The diversity preservation by the pure LC and iLaplace strategies is not sufficient to prevent premature convergence, although they are still better than the standard method. The fitness convergence is reflected also in the diversity convergence, see Figure 12.

5.3.3 LeadingOnes

The LeadingOnes problem has large plateaus consisting of all solutions with the same number of leading ones. This puts it somewhere between the flat landscape and OneMax, as it has a unique optimum, but at the same time random selection plays an important role in the plateau regions. Thus, it is important to maintain sufficient diversity in the latter bits while the front bits are being optimized. On the other hand, too much diversity is harmful at the end, as the already found leading ones are disturbed too much.

Figure 13 depicts the convergence of the different approaches. The standard update quickly converges to a fitness of only 10 (maximum is 100), which shows that LeadingOnes is a really difficult problem for EDAs. The diversity preservation of pure LC and iLaplace improve fitness somewhat, but again can not prevent premature convergence. The clear winners on this problem are the approaches that maintain a minimum diversity level, i.e. Laplace and all methods combined with BC. They all reach a fitness level of about 80, differing in the time they require to find good solutions. Overall, LC+BC seems to perform best. Furthermore, we had set the default parameters based on OneMax. However, in LeadingOnes, it is important to not disrupt the leading sequence of ones by too much exploration, thus a smaller minimal mutation rate is appropriate for this problem. When setting $\beta = 0.005$ and $\alpha = 0.05$, all the approaches maintaining a minimum diversity are able to find the optimal solution with fitness 100 within 1000 iterations (not shown).

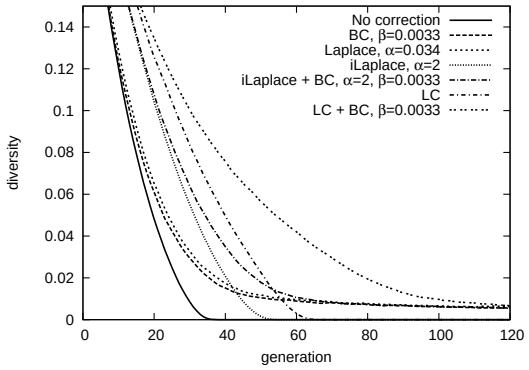

Figure 12: Comparison of the different model update strategies on the OneMax landscape, $N = 20, L = 300$.

5.3.4 NK landscapes

On the NK landscape, the different model update strategies are compared in Figure 14. As for the previous test problems, all update strategies proposed in this paper outperform the standard approach (solid line). Among the pure strategies, BC and Laplace perform very well, followed by LC and then iLaplace. However, the latter two can be combined with BC to ensure minimal mutation rates. While the combination of iLaplace with BC is still worse than BC alone, the combination of LC with BC yields the overall best results, although it converges more slowly.

6. CONCLUSION

Previous studies [4, 10, 12] have shown that the standard EDA suffers from diversity loss due to sampling errors.

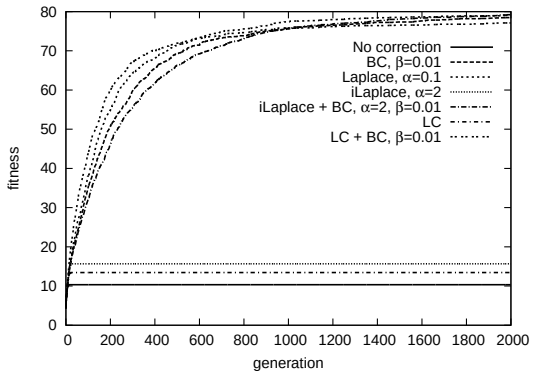

Figure 13: Comparison of the different model update strategies on the LeadingOnes landscape, best fitness found so far, $N = 20, L = 100$.

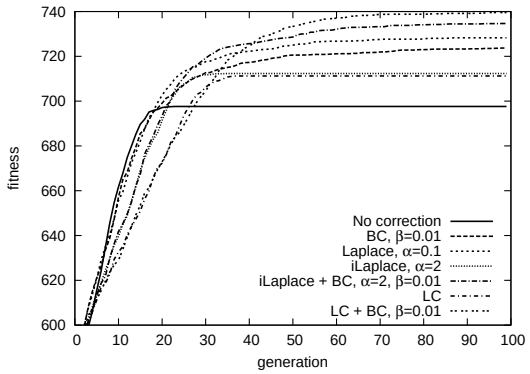

Figure 14: Comparison of the different model update strategies on the NK landscape, best fitness found so far, $N = 20, L = 50, k = 5$.

In this paper, we have proposed a number of methods to address this diversity loss.

First, we have suggested an intelligent sampling technique, called permutation sampling, which is able to completely remove the sampling error in the offspring generation phase of the algorithm. Since this method otherwise leaves the algorithm unaltered, it should always be preferred to the standard random offspring generation.

The sampling error due to selection is more difficult to address. We have suggested a number of intuitive ways to alter the model update, attempting to correct the anticipated loss, taking into account a prior distribution, or ensuring a minimal diversity level. The results in our empirical tests clearly show that all suggested approaches seem to outperform the standard update on all problems, in most cases by a huge margin. The best model update strategy depends on the application, but the idea to correct for the expected diversity loss in combination with a minimal ensured diversity level seems to preform best overall.

In the future, we plan to test our ideas more extensively also on real-world problems, and to extend the suggested methods to a larger alphabet, other selection mechanisms than truncation selection, and, more importantly, to more complex EDAs such as BOA. Finally, it seems very promising to adapt the proposed methods also on dynamic test problems, where diversity is particularly important.

7. REFERENCES

[1] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In J. Grefenstette, editor, *International Conference on Genetic Algorithms*, pages 14–21. Lawrence Erlbaum Associates, 1987.

[2] S. Baluja and S. Davies. Using optimal dependency-trees for combinatorial optimization: Learing the structure of the search space. In D. Fisher, editor, *International Conference on Machine Learning*, pages 30–38, 1997.

[3] J. Branke, M. Cutaia, and H. Dold. Reducing genetic drift in steady state evolutionary algorithms. In Wolfgang Banzhaf et al., editors, *Genetic and Evolutionary Computation Conference*, pages 68–74. Morgan Kaufmann, 1999.

[4] C. Gonzalez, J. Lozano, and P. Larrañaga. Analyzing the population based incremental learning algorithm by means of discrete dynamical systems. *Complex Systems*, 12(4):465–479, 2001.

[5] C. Gonzalez, J. Lozano, and P. Larrañaga. The convergence behaviour of the PBIL algorithm: a preliminary approach. In *International Conference on Neural Networks and Genetic Algorithms*, pages 228–231, 2001.

[6] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[7] T. Mahnig and H. Mühlenbein. Optimal mutation rate using bayesian priors for estimation of distribution algorithms. In *Stochastic Algorithms: Foundations and Applications*, volume 2264 of *LNCS*, 2001.

[8] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions i: Binary parameters. In *Parallel Problem Solving from Nature*, pages 178–187, 1999.

[9] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[10] J. L. Shapiro. Scaling of probability-based optimization algorithms. In Klaus Obermayer, editor, *Advances in Neural Information Processing Systems 15*, pages 399–406. MIT Press, 2003.

[11] J. L. Shapiro. The detailed balance principle in estimation of distribution algorhithm. *Evolutionary Computation*, 13(1):99–124, 2005.

[12] J. L. Shapiro. Diversity loss in general estimation of distribution algorithms. In *Parallel Problem Solving from Nature*, volume 4193 of *LNCS*, pages 92–101. Springer, 2006.

[13] H. Wu and J. L. Shapiro. Does overfitting affect performance in estimation of distribution algorithms. In *Genetic and Evolutionary Computation Conference*, pages 433–434. ACM, 2006.