Institut für Angewandte Informatik
und Formale Beschreibungsverfahren
Universität Karlsruhe (TH)

Forschungsgruppe E ziente Algorithmen
Prof. Dr. Hartmut Schmeck

Studienarbeit "`Reducing diversity loss in estimation of distribution algorithms"', Sommersemester 2006

Reducing diversity loss in estimation of distribution algorithms

Autor: Clemens Lode
Betreuer: J"urgen Branke

Contents

1 Introduction

Many EDAs can reach a state from which the probability of ever nding the optimum is zero. This is due to diversity loss, i.e. one or more components of all individuals in the populations become the same value. If a di erent value is required in the optimum, the optimum will never be sampled. If no action is taken to either prevent uniformity of the values in a single component or to reduce the diversity loss in each generation, the variance can never be restored or held at a certain level and simply increasing the number of generations will not increase the probabilty of nding the optimal or even a better solution. One way to counter this diversity loss is to increase the population size, by checking and correcting the populaton if uniformity in a component is reached or by using the Laplace correction. This paper will use a di erent approach by adjusting the distribution vector accordingly to the population and selection size.

2 Abstract

In [1] it was shown that the factor of diversity loss of sampling N individuals from a population of the size M and generating a new population of the size M is $1 - \frac{1}{N}$ and that this is true for a whole class of EDAs (SML-EDAs, probability model is build using only data sampled from the last generation). In this paper it will be shown that the factor of diversity loss of generating a new population of the size M on the basis of a distribution vector p is $1 - \frac{1}{M}$. Using both results we can calculate a new p (on basis of the population size M, the sampling size N and the old distribution vector) with which the diversity loss is lower. It will be demonstrated in several tests that the diversity loss is lower, that it outperforms the standard Laplace correction and that this correction not only works on at landscapes but also improves the performance in problems like OneMax or TwoPeak. The resulting method is to correct p to $\frac{1}{2}(1 - \sqrt{(\frac{N(M-1)}{M(N-1)})})$ for $p < \frac{1}{2}(1 - \sqrt{(\frac{M(N-1)}{N(M-1)})})$, to $\frac{1}{2}(1 + \sqrt{(\frac{(N(M-1)}{M(N-1)})})$ for $p > \frac{1}{2}(1 + \sqrt{(\frac{M(N-1)}{N(M-1)})}$ and to $\frac{1}{2}$ otherwise.

3 De nitions

The diversity of a given population can be measured by the 'trace of the empirical covariance matrix'.
Let

- C: number of components of each individual
- N: size of the population
- n: number of selected individuals
- A: set of di erent values a component can take
- $|A|$: number of di erent values a component can take
- x_i^μ: value of component i of individualμ
- $\varphi(x)$: '1' if the condition x is met, '0' otherwise

v_i^a describes the ratio of a certain value of a component in the population. E.g. $v_2^0 = 0.3$ means that 30% of all the individuals in the population have a '0' in the second component.

$$v_i^a = \frac{1}{N} \sum_{\mu=1}^{N} \varphi(x_i^\mu = a) \tag{1}$$

v_i denotes the average variance of all values of a certain value of a component in the population.

$$v_i = \frac{1}{|A|} \sum_{a=0}^{|A|-1} v_i^a (1 - v_i^a) \tag{2}$$

v denotes the sum of the average variances of all values of the values in all components in the population.

$$v = \frac{1}{|A|} \sum_{i=1}^{C} \sum_{a=0}^{|A|-1} v_i^a (1 - v_i^a) \tag{3}$$

The scope of this paper are UMDAs and apply them on problems with a at tness landscape and problems where the components are not interconnected (e.g. OneMax). Therefor we can examine the variance of each component independently, i.e. ~~Set~~ 1.

Creating a population with the size of M with $|A|$ di erent values in each component we then have a combination with repetition, i.e. there would be

$$\frac{(|A|+M-1)!}{M!(|A|-1)!} = \binom{|A|+M-1}{M} \tag{4}$$

possible di erent population. For example for $|A| = 2$ there would be $M+1$ di erent populations (...000, ...001, ...011, ...111, ...) as the position of an individual in the population is not important. Although there are implementations of UMDA with $|A| > 2$ (see TODO) we will only look into the case of $|A| = 2$, i.e. our UMDAs are represented as bitstrings.

With UMDAs we determine in each step the ratio of '1' for each component of the selected part of the population and calculate the ratio. In the literature (see TODO) this is usually called p_i and it is equal to our de nition of $v_i^{a=1}$, so we set $p_i := v_i^1$. The distribution vector is called $p := p_1, p_2, ..., p_C$. From this distribution vector p a new population of the size of M is generated where each individual has a '1' in its ith component with the probability p_i (or likewise a '0' with the probability of $(1-p_i)$). The distribution vector of the next generation will be called $\tilde{p}$.

The probability for $\tilde{p}_i = \frac{k_i}{M}$, i.e. the generation of a population with k_i '1's in the ith component on the basis of a given distribution vector p, is

$$P(\tilde{p}_i = \frac{k_i}{M}) = p_i^{k_i}(1-p_i)^{M-k_i}\binom{M}{k_i} \tag{5}$$

As de ned above, v denotes the sum of the average variances of all values of the values in all components in the population. In connection to a given population with $|A| = 2$ and with $k = k_1, k_2, ..., k_C$ with k_i denoting the number of '1's in the ith component, we get

$$v_k = \frac{1}{|A|=2}\sum_{i=1}^{C}\sum_{a=0}^{|A|-1=1} v_i^a(1-v_i^a) \tag{6}$$

For a given k we can calculate the v_i^as

$$v_i^{a=0} = \frac{M-k_i}{M}$$

$$v_i^{a=1} = \frac{k_i}{M}$$

This is true because $\varphi(x_i^\mu = 1) = k$ and $\varphi(x_i^\mu = 0) = (M-k)$.

So we get:

$$v_k = \frac{1}{2}\sum_{i=1}^{C}\sum_{a=0}^{1} v_i^a(1-v_i^a) =$$

$$\frac{1}{2}\sum_{i=1}^{C}[v_i^0(1-v_i^0)+v_i^1(1-v_i^1)]=$$

$$\frac{1}{2}\sum_{i=1}^{C}[\frac{M-k_i}{M}(1-\frac{M-k_i}{M})+\frac{k_i}{M}(1-\frac{k_i}{M})]=$$

$$\sum_{i=1}^{C}\frac{k_iM-k_i^2}{M^2}$$

And

$$v_{k_i}=\frac{k_iM-k_i^2}{M^2} \tag{7}$$

Now we have on the one side the probability for the generation of a population with a certain k and a given$p$ and the variance of such a population. The expected variance d_p is therefor the sum of the product of the variance and the probability for/of all values of k_i:

$$d_p=\sum_{i=1}^{C}\sum_{k_i=0}^{M}v_{k_i}P(\tilde{p}_i=\frac{k_i}{M})=\sum_{i=1}^{C}\sum_{k_i=0}^{M}\frac{k_iM-k_i^2}{M^2}p_i^{k_i}(1-p_i)^{M-k_i}\binom{M}{k_i} \tag{8}$$

4 Preventing variance loss TODO

In the scope of this paper (UMDAs) each component is independent, so we can examine each single component on its own and set $\ell = 1$ (and $k = k_{i=1}$ and $p = p_{i=1}$ for simplicity).

From [1] we know that the variance loss of the three steps (selecting N individuals, calculating a distribution vector p and sampling M individuals) is $1 - \frac{1}{N}$, i.e. (with v_t denoting the variance of the population of last generation of which we selected N individuals v_{t+1} denoting the variance of the current population)

$$v_{t+1} = (1 - \frac{1}{N})v_t \tag{9}$$

1. Population of the previous generation $\Rightarrow$ Variance v_t
2. Select N individuals and calculate p
3. Generate new population of size M on basis of $p \Rightarrow$ Variance $v_{t+1} = (1 - \frac{1}{N})v_t$
4. done.

When generating a new population on basis of p we learned last section that the variance is d_p. But the variance that we want is $p(1-p)$, i.e. the variance of a population of in nite size generated on basis of p. Thus the factor of variance loss from step 3 to 4 is $\frac{d_p}{p(1-p)}$. Using for example the math program Maple (which uses hypergeometric series) this can be simpli ed to:

$$\frac{d_p}{p(1-p)} = \frac{1}{p(1-p)} \sum_{k=0}^{M} \frac{kM - k^2}{M^2} p^k (1-p)^{M-k} \binom{M}{k} = 1 - \frac{1}{M} \tag{10}$$

So we have:

- Factor of variance loss from step 3 to 4 $1 - \frac{1}{M}$
- Factor of Variance loss from step 2 to 4 $1 - \frac{1}{N}$

So our factor y of variance loss from step 2 to step 3 is:

$$1 - \frac{1}{N} = y \frac{\dot{d}_p}{p(1-p)} \Leftrightarrow y = \frac{1 - \frac{1}{N}}{\frac{\dot{d}_p}{p(1-p)}} \tag{11}$$

With equation (10) we get $y = \frac{1-\frac{1}{N}}{1-\frac{1}{M}} = \frac{M(N-1)}{N(M-1)}$.
If we use a di erent distribution vector q to generate the population and set $q(1-q)$ (or q accordingly) in a way so that the theoretical factor of variance loss factor is 1 (i.e. no variance loss) we will reduce the overall variance loss. As our variance loss is $y = \frac{M(N-1)}{N(M-1)}$ we have to multiply with the reciprocal value in order to nullify the variance loss and multiply that factor with our original $p(1-p)$ variance. We are allowed to do that because the variance loss itself does not depend on the distribution vector we use:

$$x = \frac{1}{y} = \frac{N(M-1)}{M(N-1)} \tag{12}$$

$$q(1-q) = p(1-p)x \Leftrightarrow -q^2+q-(-p^2+p)x = 0 \Leftrightarrow q_{1/2} = \frac{1}{2}(1 \pm \sqrt{1-4(-p^2+p)x}) \tag{13}$$

For example with $M = 2N$ (i.e. we select half of the population to generate a new distribution vector) we get

$$q_{1/2} = \frac{1}{2}(1 \pm \sqrt{1-4(-p^2+p)\frac{N(2N-1)}{2N(N-1)}}) =$$

$$\frac{1}{2}(1 \pm \sqrt{1-4(-p^2+p)\frac{2N-1}{2N-2}})$$

For $1-4(-p^2+p)\frac{N(M-1)}{M(N-1)} < 0$ we get a negative value in square root. The border values for p are:

$$1-4(-p^2+p)\frac{N(M-1)}{M(N-1)} = 0 \Leftrightarrow$$

$$p_{1/2} = \frac{1}{2}(1 \pm \sqrt{1-\frac{M(N-1)}{N(M-1)}})$$

In gure 1 you can see the graph of the correction function for this case for various population sizes. If our p is within p_1 and p_2

$$\frac{1}{2}(1-\sqrt{1-\frac{M(N-1)}{N(M-1)}}) < p < \frac{1}{2}(1+\sqrt{1-\frac{M(N-1)}{N(M-1)}})$$

we will have to substitute with an appropriate value. As the function approaches 0.5 both from the top and from the bottom we will substitute with $q = 0.5$ if our p is within those borders.

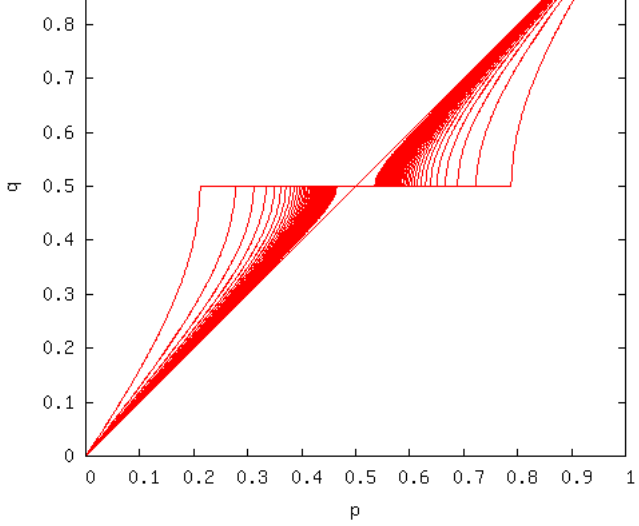


Figure 1: For a set of values for n (with population size $n = 2N$) ranging from 4 (farthest on the left and right) to 200 (nearest to $f(p) = p$)

5 Code

Finally we can put our results into a code ($M = 2N$):

```
if( p < 0.5 * (1 - sqrt( (2*N-2) / (2*N-1) )) )
     q = 0.5 * (1 - sqrt( (2*N-1) / (2*N-2) * 4*p*(1-p) ) );
else
if( p > 0.5 * (1 + sqrt( (2*N-2) / (2*N-1) )) )
     q = 0.5 * (1 + sqrt( (2*N-1) / (2*N-2) * 4*p*(1-p) ) );
else
     q = 0.5;
```

or for random values of population size M and sampling size N:

```
if( p < 0.5 * (1 - sqrt( (M*(N-1)) / (N*(M-1)) ) ) )
     q = 0.5 * (1 - sqrt( (N*(M-1)) / (M*(N-1)) * 4*p*(1-p) ) );
else
if( p > 0.5 * (1 + sqrt( (M*(N-1)) / (N*(M-1)) ) ) )
     q = 0.5 * (1 + sqrt( (N*(M-1)) / (M*(N-1)) * 4*p*(1-p) ) );
else
     q = 0.5;
```

It is easy to implement the code piece within an existing application as it is problem independent (within the restrictions described above, bitstrings on a flat landscape in UMDA). The code itself has to be inserted just after determining the distribution p from the selected part of the population of the size N.

6 Options

6.1 Exact Distribution

We have calculated that the variance loss of generating a population on basis of p randomly will result in a variance loss of $1 - \frac{1}{M}$. In order to create a new generation we originally set each value of each component of an individual of the new population to '1' with the probability of p (or to '0' with the probability of $1 - p$).
A different approach is to create the new generation by distributing exactly $p * M$ '1's (or $(1 - p) * M$ '0's) within the array of components in the population so that the new variance is exactly $p(1 - p)$, i.e. there is no variance loss in that step. We will call this Exact Random Distribution .

$$p = \frac{k_{old}}{N} k_{new} = \lfloor p * M \rfloor = \lfloor \frac{k_{old} * M}{N} \rfloor$$

6.1.1 Rounding errors

Rounding errors are a big problem as with iterative algorithms like EDAs they will multiply over time. If an algorithm is for example biased towards '1' it will do great in a simple OneMax test but fail miserably in a 'ZeroMax' test.
When we analyze this option within our framework that we have set in the last two sections we see that a rounding error occurs when $p * M$ is no whole number. When we simply calculate p by counting the number of '1's in the selected part of the population of size N such a rounding error will only occur if M is no multiple of N. No problem here, but this becomes an issue if we try to combine for example the Laplace correction or the distribution correction with Exact Distribution, because $p * M$ is most likely not a whole number, no matter whether M is a multiple of N or not.
Thus, when using Exact distribution, we can either demand that $\lfloor p * M \rfloor = p * M$ or we change the method so that $p * M$ '1's are created. This can be done by using the remainder as the probability to add another '1' to the population. For example if $p = 0.71$ and $M = 10$ we will distribute seven '1's and an eighth '1' with the probability of 0.1. The drawback is that using this method causes the distribution no longer deserving the title 'exact'.

6.1.2 Exact Distribution Correction

Assuming we Even more important are the changes in our definitions and equations. In equation (5) the probability for $\tilde{p_i} = \frac{k_i}{M}$ obviously changes as there are always exactly $[p * M] + (p * M - [p * M])$ '1's in the population, i.e.

$$\dot{P}(\tilde{p_i} = \frac{k_i}{M}) = \begin{cases} 1 & \text{for } k_i = p_i * M \\ 0 & \text{for } k_i \neq p_i * M \end{cases}$$

The other equations about our variance v_{k_i} remain the same as they are dependent on our k_i anyways. Changes have to be made to equation (8) because $P(\tilde{p_i} = \frac{k_i}{M})$ has

changed. We no longer sum over all possible values of k as only value ($k = p * M$) has the probability '1' and all other summands the probability '0':

$$\dot{d}_p = \sum_{i=1}^{C} v_{k_i=p_i*M} P(\tilde{p}_i = \frac{k_i = p_i * M}{M}) = \sum_{i=1}^{C} v_{p_i*M} * 1 = \sum_{i=1}^{C} \frac{p_i * M^2 - (p_i * M)^2}{M^2} = \sum_{i=1}^{C} p_i * (1 - p_i) \tag{14}$$

As expected we have no loss of variance from step 3 to step 4 using exact random distribution because $\frac{\dot{d}_p}{p(1-p)} = 1$ and we get in an analogous manner the factor of variance loss from step 2 to 3 with $y = 1 - \frac{1}{N}$ and $x = \frac{1}{1-\frac{1}{N}} = \frac{N}{N-1}$ which we can insert into equation (13). There we get $\dot{q}(1-q) = p(1-p) * \frac{N}{N-1}$ and we have

$$\dot{q}(1 - \dot{q}) = p(1-p)\frac{N}{N-1} \Leftrightarrow$$

$$-\dot{q}^2 + \dot{q} - (-p^2 + p)\frac{N}{N-1} = 0 \Leftrightarrow$$

$$\dot{q}_{1/2} = \frac{1}{2}(1 \pm \sqrt{1 - 4(-p^2 + p)\frac{N}{N-1}})$$

For $1 - 4(-p^2 + p)\frac{N}{N-1} < 0$ we get a negative value in square root. The border values for p are:

$$1 - 4(-p^2 + p)\frac{N}{N-1} = 0 \Leftrightarrow$$

$$p_{1/2} = \frac{1}{2}(1 \pm \sqrt{1 - \frac{N-1}{N}})$$

This method will be called Exact Distribution Correction . Compared

Comparison : no correction / exact random distribution / correction / exact random distribution correction + exact random distribution / correction + exact random distribution

Directly comparing 'exact random distribution' with

Even more important are the changes in our de nitions and equations. In equation (5) the probability for $\tilde{p}_i = \frac{k_i}{M}$ obviously changes as there are $\lfloor p * M \rfloor$ '1's with the probability $1 - (p * M - \lfloor p * M \rfloor)$ and $\lfloor p * M \rfloor + 1$ '1's with the probability $p * M - \lfloor p * M \rfloor$.

$$\dot{P}_(\tilde{p}_i = \frac{k_i}{M}) = \begin{cases} 1 - (p * M - \lfloor p * M \rfloor) & \text{for } k_i = \lfloor p_i * M \rfloor \\ p * M - \lfloor p * M \rfloor & \text{for } k_i = \lfloor p_i * M \rfloor + 1 \\ 0 & \text{else} \end{cases}$$

The other equations about our variance v_{k_i} remain the same as they are dependent on our k_i anyways. Changes have to be made to equation (8) because $P(\tilde{p}_i = \frac{k_i}{M})$ has changed and we no longer sum over all possible values of k_i but only two:

$$\dot{d}_p = \sum_{i=1}^{C} \sum_{k_i=0}^{M} v_{k_i} P(\tilde{p}_i = \frac{k_i}{M}) =$$

$$\sum_{i=1}^{C} v_{k_i=\lfloor p_i*M \rfloor} P(\tilde{p_i} = \frac{k_i = \lfloor p_i * M \rfloor}{M}) + v_{k_i=\lfloor p_i*M \rfloor+1} P(\tilde{p_i} = \frac{k_i = \lfloor p_i * M \rfloor + 1}{M}) =$$

$$\sum_{i=1}^{C} v_{\lfloor p_i*M \rfloor}(1 - (p * M - \lfloor p * M \rfloor)) v_{\lfloor p_i*M \rfloor+1}(p * M - \lfloor p * M \rfloor) =$$

$$\sum_{i=1}^{C} \frac{\lfloor p_i * M \rfloor M - \lfloor p_i * M \rfloor^2}{M^2}(1-(p*M-\lfloor p*M \rfloor)) + \frac{(\lfloor p_i * M \rfloor + 1)M - (\lfloor p_i * M \rfloor + 1)^2}{M^2}(p*M-\lfloor p*M \rfloor) =$$

$$\sum_{i=1}^{C} \frac{(\lfloor p_i * M \rfloor)^2 - 2 * \lfloor p_i * M \rfloor * p * M + p * M^2 - p * M + \lfloor p * M \rfloor}{M^2}$$

Our variance loss is as before $\frac{\dot{d}_p}{p*(1-p)}$ which unfortunately cannot simpli ed any further.

With equation (11) we have $y = \frac{1-\frac{1}{N}}{\frac{\dot{d}_p}{p*(1-p)}}$ and $x = \frac{1}{y} = \frac{\frac{\dot{d}_p}{p*(1-p)}}{1-\frac{1}{N}}$. The problem is that we cannot easily insert that into equation (13) as in this case our depends on the distribution vector with which we create the new population. So we really have to make x dependent on the distribution vector $\dot{q}$:

$$\dot{q}(1 - \dot{q}) = p(1 - p)x_q$$

TODO This is kind of di cult to solve for q.

For $1 - 4(-p^2 + p)\frac{N}{N-1} < 0$ we get a negative value in square root. The border values for p are:

$$1 - 4(-p^2 + p)\frac{N}{N-1} = 0 \Leftrightarrow$$

$$p_{1/2} = \frac{1}{2}(1 \pm \sqrt{1 - \frac{N-1}{N}})$$

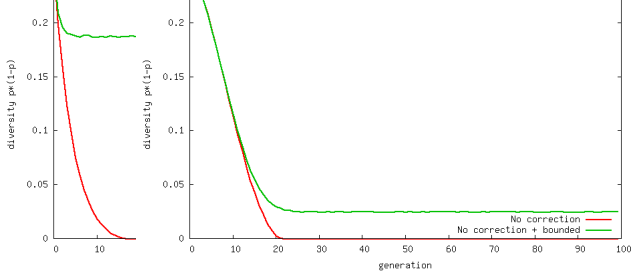
This method will be called Exact Distribution Correction . Compared

7 Boundary check

Another option is called bounded}. If it is set the boundaries of the nal p are checked. In the case p gets above $1 - \frac{1}{M}$ or below $\frac{1}{M}$ it is corrected to these boundaries:

$$p = \begin{cases} \frac{1}{M} & \text{for } p < \frac{1}{M} \\ 1 - \frac{1}{M} & \text{for } p > 1 - \frac{1}{M} \\ p & \text{else} \end{cases}$$

This is only useful for some of the correction algorithms that will stuck when reaching values near 0 or 1 for p, i.e. if the selected part of the population contains only '0's or '1's in any component and the algorithm calculated $p = 0.0$ or $p = 1.0$. The boundary check theoretically ensures that



' check

Figure 2: With boundary check the diversity is held at a constant level

any algorithm will nd the optimal solution eventually while reducing the convergence speed because values that are already determined to be optimal can be changed. In practice this is of not much use if it is the only method to prevent diversity loss but it might increase the e ectivity of some of the algorithms if the algorithm is allowed to run a very long time, see section Tets for a comparison. Generally speaking adding the bounded option to an algorithm will reduce its convergence speed signi cantly but will catch up and even surpass the original algorithm several generations later depending on the parameters and the problem. TODO kuerzen

An issue which will not be discussed here is that when using boundary check, our diversity will change and we would need to adapt our correction formulas accordingly if we want to use them in connection with boundary check. TODO

You can see here that with boundary check the diversity is held at a constant level. For each component there is always at least a chance of $\frac{1}{M}$ to switch between '0' and '1'. On the one side this has the positive e ect that theoretically the optimal solution will be found. On the other hand single components that already contain an optimal value can switch back to the wrong value. TODO

TODO Vergleiche innerhalb von Gruppen, also No correction No correction+bounded, diversity correction mit diversity correction+bounded etc.

For onemax we clearly see that these two opposite e ects let the function converge at a lower level:

TODO While it has a chance to nd a better solution (in contrary to the method without correction) it has trouble keeping up the correct values that were found so far.

TODO With higher population sizes this e ect vanishes as $\frac{1}{M}$ becomes smaller.

7.1 Laplace

Laplace correction is more or less the standard (TODO Quelle) in connection with UMDA. On the one side it ensures that no component gets stuck at $p = 1.0$ or $p = 0.0$ and on the other hand it biases the resulting p towards $\frac{1}{2}$. The general formula for Laplace correction for a component i is

$$p_i = \frac{k_i + \alpha}{M + 2\alpha} \tag{15}$$

TODO

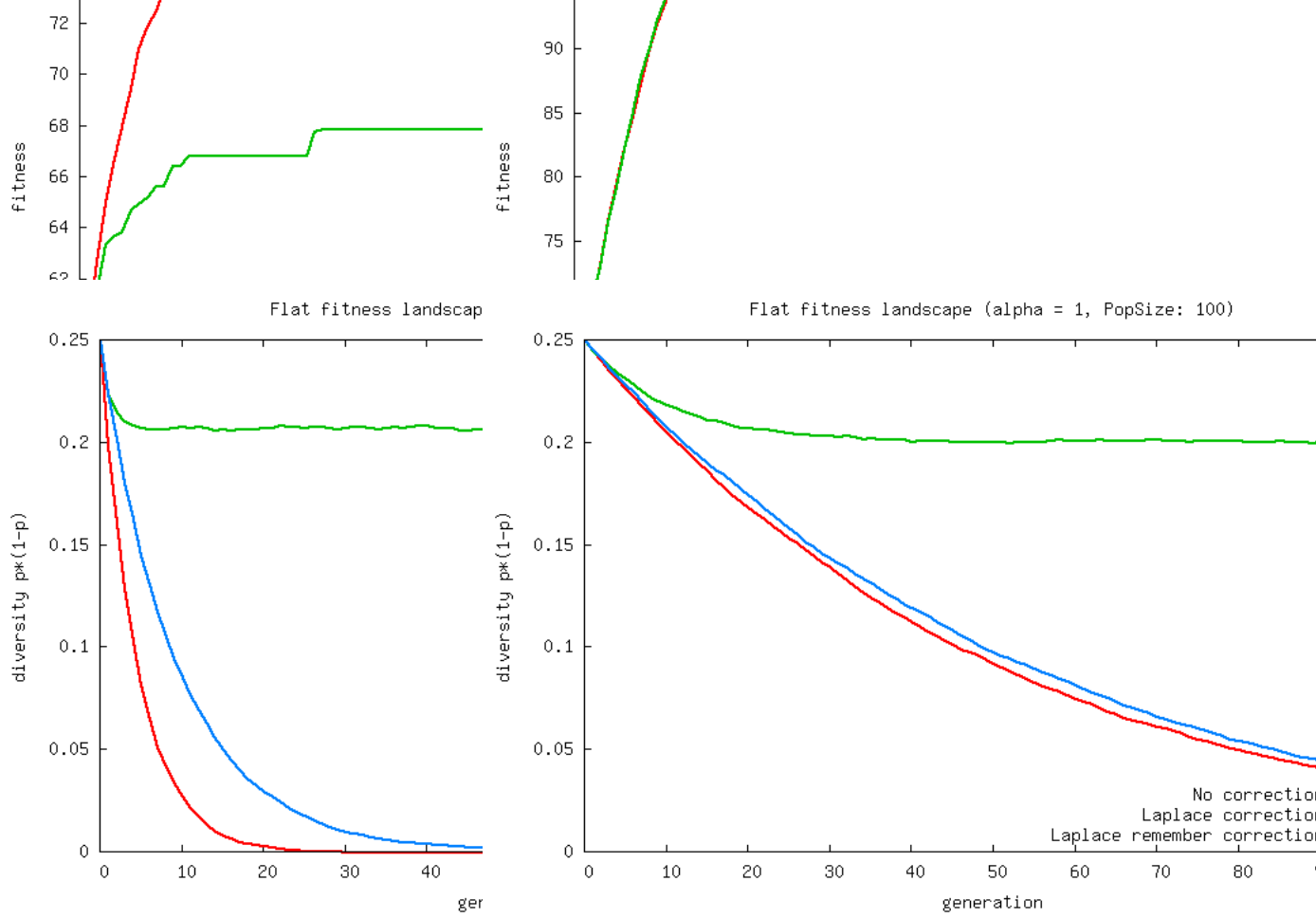


Figure 4: blabla TODO

Optionally we also look at the case where α is the distribution vector of the previous generation, i.e.

$$\tilde{p}_i = \frac{k_i + p_i}{M + 2}$$

where $\tilde{p}$ denotes the new and p denotes the old distribution vector. Here we have of course no boundary check as with a constant α. We now will compare this method with Laplace with $\alpha = 1.0$ (other values of α simply in- or decrease the constant level of variance).

Looking at the variance on a at tness landscape, we can clearly see that using a the previous distribution vector p as alpha does not prevent diversity loss. TODO

In the case of ONEMAX: TODO - outperform laplace - higher population size - as usual - less advantage over no correction

With this parameter settings and problem, higher values for α will decrease the e ciency further:

7.2 Misc

The remaining methods are

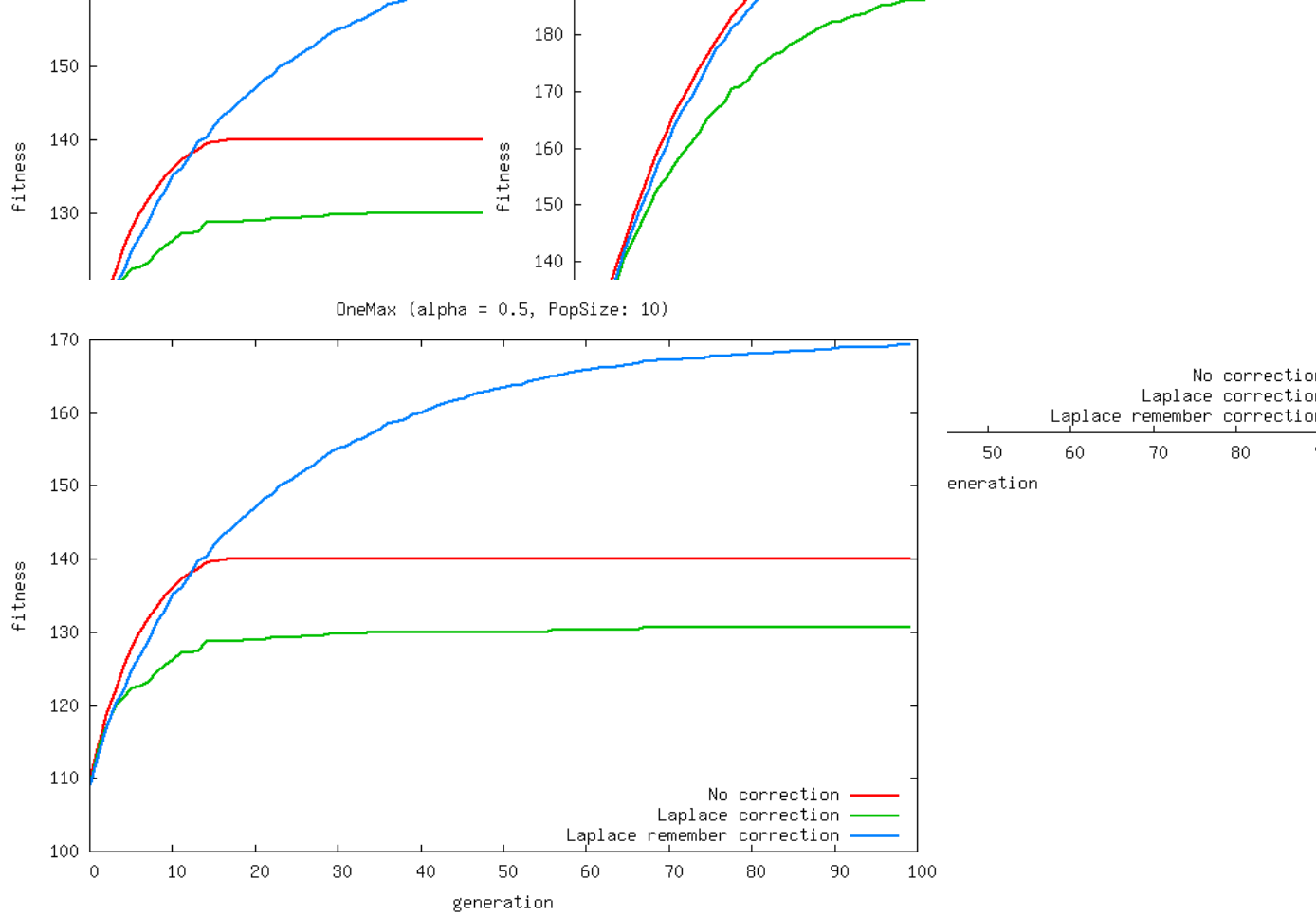


Figure 6: blabla TODO

1. No correction: The new distribution p for each component is simply calculated by dividing the number of '1's by the sample size
2. Corrected distribution: Same as (1), but the resulting p is corrected with the formula discussed in the previous section

8 General test con guration

8.1 Graph drawing

Because all algorithms are based on random numbers 50 seperate runs were made for each parameter setting and algorithm. The data then is averaged and drawn as a graph. Normally this would result in oscillating tness graphs as shown for example in gure (7). The oscillation does not bear any additional information so for the sake of clarity only the best average result that was found until that point is drawn, i.e. the graphs always have a non-negative gradient. In gure (7) you can also see that Corrected distribution and No correction do not oscillate. This is simply because their variance dropped to 0 after a few generations, i.e. no component will change anymore.

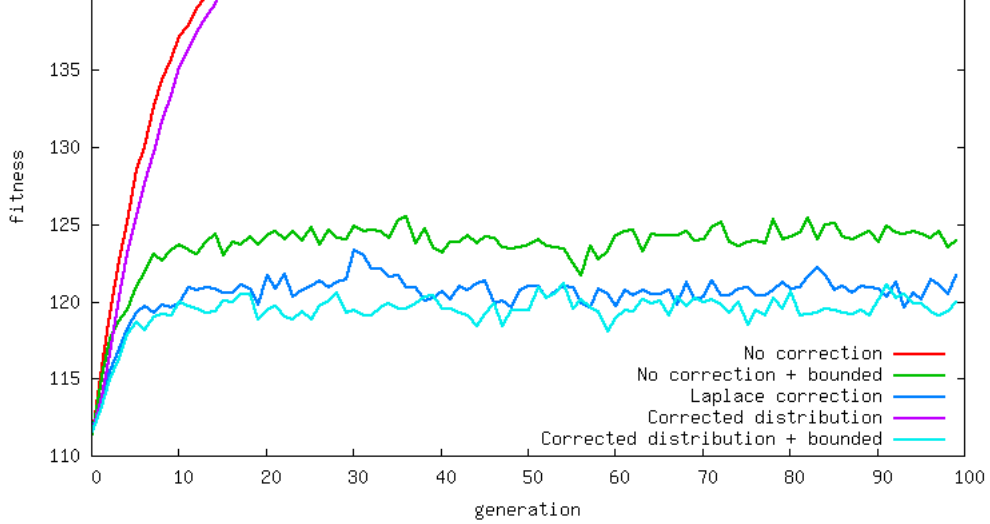


Figure 7: Not preventing oscillating of tness graphs TODO

The graphs were created with the help of gnuplot 4.0 and g++ 4.0 (gnu C++), the actual program will be availible with a later version of this paper on CD-ROM.

8.2 Population size

Depending on the problem type, problem size and algorithm di erent population sizes ranging from 10 to 100 were tested.

As the convergence speed di er between the methods (dependent on population size, problem type and problem size) several di erent population sizes were tested.

Population size

TODO vielleicht noch irgendwie Zeit messen

9 Test types

9.1 Flat tness landscape

In this test we examine the behaviour of the algorithms in terms of their diversity with varying parameters on a at tness landscape. It is basicly a 'needle in a haystack' problem where the needle is not found within the 200 generations, i.e. all solutions have the same tness. The problem size is a bitstring with length 10, i.e. we have 10 components.
The additional graph, '1 - 1/N loss / generation', denotes the theoretical loss of diversity according to [1].

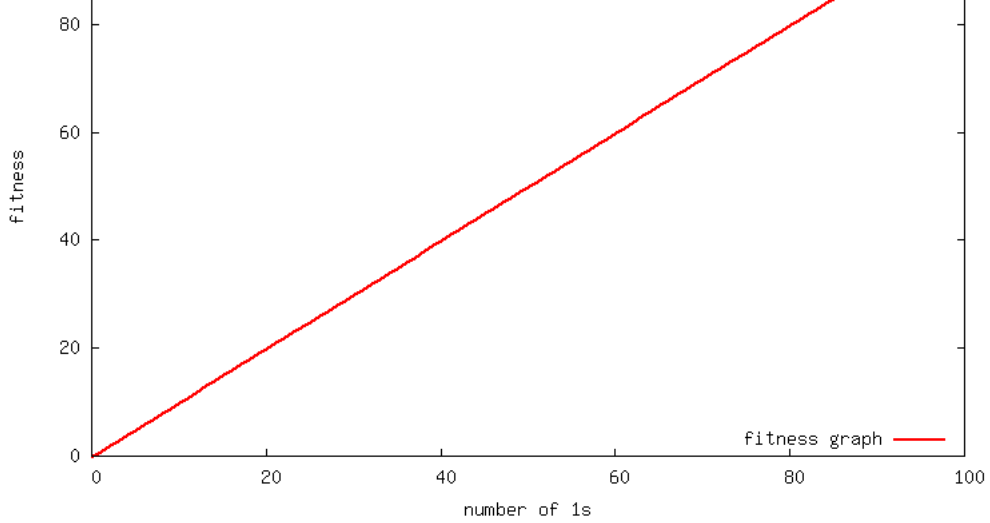


Figure 8: Fitness Landscape of OneMax

9.2 OneMax problem

In this test we examine the behaviour of the algorithms in terms of their tness with varying parameters with the problem OneMax. The goal is to nd the string '111...', each '1' in a component of an individual gets rewarded by one tness point. In our test the problem size ranges from 100 to 500 depending on what is wanted to be demonstrated.
For each generation only the highest tness of each population is taken into account. Again the graph shows the average of the tness values of all runs.
TODO

TODO With the tests we have shown that our 'Corrected distribution' algorithm does signi cantly reduce the diversity loss compared to an algorithm with no correction. Making sure that the algorithm does not stall in a component with $p < \frac{1}{\tilde{n}}$ or $p > 1 - \frac{1}{\tilde{n}}$ ("bounded") makes sure that the diversity does not drop to zero in the long run. 'Corrected distribution' does not outperform 'Laplace correction' as the latter always corrects any distribution towards $p = 0.5$.

TODO di erence - selecting!

While the OneMax problem does not represent a problem with a real at tness landscape (thus the theory we discussed earlier does not apply here), a local at tness landscape can occur if the variance drops and/or the tness values are very similar (e.g. 010, 100, 001).
'Corrected distribution + Laplace' is clearly worse than 'Laplace correction', the changes made to the distribution p (that we determined on base of the selected individuals) are too big in order to get useful results. As expected 'Laplace correction' itself does not score well, it converges at a signi cant lower tness level.
'Exact random distribution' seems to generally improve the convergence of methods using not the 'Laplace correction' while being indi erent or being even worse for the 'Laplace correction' itself.
At least for the case of 'OneMax' and the given parameter con guration, 'Corrected distribution' in connection with 'Exact Random Distribution'

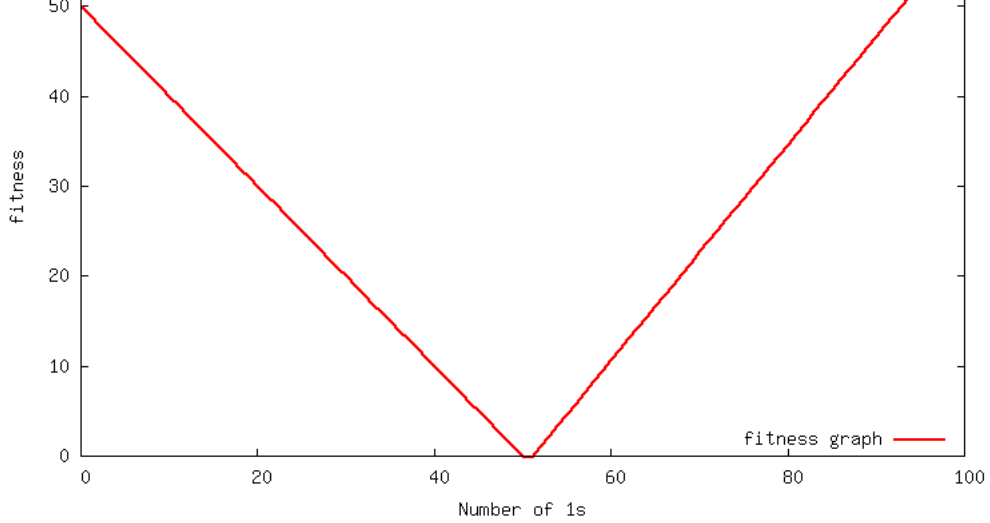


Figure 9: Fitness landscape of OneMax with two peaks

seems to outperform all other methods.

9.3 TwoPeak problem

Similar to the OneMax problem we do count the number of equal values but not only '1's but '0's, too. There are two parameters, one de nes the center where we switch from counting '0's to counting '1's and one is a factor which determines how much more (or less) worth is a '1' compared to a '0'.

6 di erent algorithms were tested:

9.4 Leading-1s problem

In the Leading-1s problem the tness is calculated by counting the number of leading '1's. This implies that contrary to OneMax the bitposition is important and the bits are connected, i.e. component i's tness depends on all components $j < i$. This is more di cult to handle for UMDA algorithms discussed in this paper. Two abilities are tested with this problem. On the one side keeping correct solutions on lower bitpositions is very important, i.e. lower bitpositions contribute to the total tness more than higher bitpositions. Any algorithm that keeps up a high diversity in positions that already have the correct value will have a lower tness.

TODO BILD

9.5 Dingens Problem

TODO

Using the newly calculated distribution p_i for each component i we generate the new population of the size M depending on our option called Exact Random Distribution. If it is not set we create each member of the population individually, if it is set we distribute $p * M$ '1's and $(1-p)M$ '0's between the individuals.

10 Fields of further research

According to [1] the diversity loss of $1 - \frac{1}{n}$ is independent of $|A|$ and is valid for a whole class of so-called SML-EDAs, including UMDA, MIMIC, FDA or BOA. In this paper I have assumed that $|A| = 2$, i.e. that a component can only take two di erent values. With values $|A| > 2$ the proof has to be adapted from equation (5) on, depending on how one creates such a population. It also remains to be investigated how a similar correction of the diversity loss is possible with other methods than UMDA that fall into the category of the SML-EDAs. It is probable that the idea itself, i.e. determining the loss of variance due to sampling and calculating it backwards in order to correct the distribution p, seems to be applicable to many di erent forms of SML-EDAs or even EDAs in general - assuming one can calculate the actual diversity loss.

References

[1] Shapiro, J.L.: Diversity loss in general estimation of distribution algorithms , 2006.