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"

von Clemens Lode

eingereicht am 15.11.2006
beim Institut für Angewandte Informatik
und Formale Beschreibungsverfahren (AIFB)
der Universität Karlsruhe (TH)

Betreuer: Jürgen Branke
Referent: Prof. Dr. H. Schmeck

Heimatanschrift:
Edelweiÿweg 16
87493 Lauben
clemens@lode.de

Studienanschrift:
Lötzener Straÿe 16
76139 Karlsruhe

Ich versichere hiermit wahrheitsgemäÿ, die Arbeit bis auf die dem Aufgabensteller bereits bekannte Hilfe selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, den 15. November 2006

Contents

1 Introduction

Many Estimation of Distribution Algorithms (EDAs) can reach a state where the probability of ever finding the optimum is zero which is due to diversity loss. Diversity of a population denotes the ratio of different values in the population, i.e. if uniformity in all components is reached the diversity of this population is zero. If a different set of values is required for the optimal solution 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 diversity can never be restored or held at a certain level and continuing the calculation will not increase the probability of finding the optimal or even a better solution.
In [1] it was shown that the expected factor of diversity loss of selecting N individuals from a population of the size M and generating a new population of the size M is $1 - \frac{1}{N}$ on a flat fitness landscape. It was shown that this is true for a whole class of EDAs where the distribution vector p is build using only data sampled from the last generation, the population and selection size M and N is constant and the new generation is created randomly on basis of the distribution vector.
One way to counter this diversity loss is to increase the population size, by checking and correcting the population if uniformity in a component is reached or by using the Laplace correction. This report will use a different approach by adjusting the distribution vector according to the population size, selection size and generation method.
In this report it will be shown that the factor of diversity loss of randomly generating a new population of the size M is $1 - \frac{1}{M}$ and it will be calculated that the diversity loss of selecting N individuals can be reverted with the factor $x = \frac{N(M-1)}{M(N-1)}$ and it will be shown that the resulting correction method improves the fitness and diversity on different fitness landscapes.
In addition, instead of randomly creating a new generation by applying the distribution vector on each single component of all individuals of the population, distributing exactly $p \cdot M$ '1's in the population will result in a total loss of diversity of only $\frac{M(N-1)}{N(M-1)}$ instead of $1 - \frac{1}{N}$. It will be shown that reverting the loss of diversity will result in better performance with complex problems. The resulting method corrects p to $\frac{1}{2}(1 - \sqrt{1-x})$ for $p < \frac{1}{2}(1 - \sqrt{1 - \frac{1}{x}})$, to $\frac{1}{2}(1 + \sqrt{1-x})$ for $p > \frac{1}{2}(1 + \sqrt{1 - \frac{1}{x}})$ and to $\frac{1}{2}$ otherwise.
In this report at first we will define the expected diversity of a population (Chapter 2), then calculate the loss of diversity of selecting N individuals and generating M individuals and how to revert it (Chapter 3), then examine the distribution of exactly $p \cdot M$ '1's in the population and how rounding errors are a problem. In the second part four fitness landscapes are defined (Chapter 5), optional methods that prevent p from containing extreme values are discussed (Chapter 6) and the test configuration is explained (Chapter 7). Finally various methods are compared and discussed (Chapter 8) and the results concluded (Chapter 9).

2 De nitions

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 individual μ
- $\varphi(x)$: '1' if the condition x is met, '0' otherwise

2.1 Variance of a population

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 report is to apply Univariate Marginal Distribution Algorithms (UMDA s) on problems with a at tness landscape. UMDAs assume independence between variables, i.e. we examine each component independently and set $C = 1$. In addition the UMDAs we will discuss here are represented as bit-strings, i.e. $|A| = 2$.

As de ned above, v denotes the sum of the average variances of all values of all components in the 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 for equation 3:

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

and for a given k we can calculate the v_i^a's

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

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

This is true because $\sum_{\mu=1}^{M} \varphi(x_i^\mu = 1) = k_i$ and $\sum_{\mu=1}^{M} \varphi(x_i^\mu = 0) = (M - k_i)$.

So we get:

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

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

And

$$v_i = \frac{k_i M - k_i^2}{M^2} \tag{4}$$

2.2 Random Distribution with UMDAs (RD)

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 [2]) 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 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 $(1 - p_i)$). The distribution vector of the next generation will be called p_{t+1} and this way of creating a new generation will be called Random Distribution (RD). In chapter 4 we will look into a di erent method, called Exact Distribution (EC), which distributes exactly $p \cdot M$ '1's within the population. If no correction method (correction methods will be discussed later) is applied, it will simply be called No Correction (NoC).

The probability for $p_{t+1;i} = \frac{k_i}{M}$, i.e. the probability for generating a population with k_i '1's in the ith component on the basis of a given distribution vector p, is

$$P(p_{t+1;i} = \frac{k_i}{M}) = p_{t;i}^{k_i}(1 - p_{t;i})^{M-k_i}\binom{M}{k_i} \tag{5}$$

2.3 Expected Variance of a population

We have on the one side the probability for the generation of a population with a certain k and a given p (equation 5) and the variance of such a population (equation 4). The expected variance d_p of the whole population on the basis of p and M is therefore the sum over all products of the variance and the probability of all values of k_i:

$$d_p = \sum_{i=1}^{C}\sum_{k_i=0}^{M} v_{k_i} P(p_{t+1;i} = \frac{k_i}{M}) = \sum_{i=1}^{C}\sum_{k_i=0}^{M} \frac{k_i M - k_i^2}{M^2} p_i^{k_i}(1 - p_i)^{M-k_i}\binom{M}{k_i} \tag{6}$$

3 Preventing variance loss with Random Distribution Correction (RDC)

We will now examine a single component and how to prevent some of the variance loss that happens from one generation to the next. As previously stated in the scope of this report (UMDAs) each component is independent, i.e $C = 1$ and we set $k = k_{i=1}$ and $p = p_{i=1}$ for simplicity. The creation of one generation is as follows:

Given: Population of the previous generation⇒Variance v_t

(I) Select N individuals and calculate p

(II) Generate new population of size M on basis of p ⇒Variance v_{t+1}

3.1 Variance loss

From [1] we know that the variance loss of the two steps on a flat landscape with Random Distribution is $1 - \frac{1}{N}$, i.e. (with v_t denoting the variance of the population of the last generation from which we have selected N individuals and v_{t+1} denoting the variance of the current population)

$$v_{t+1} = (1 - \frac{1}{N})v_t \tag{7}$$

When generating a new population on basis of p we learned in the last chapter (equation 6) that the total variance of the population that we will have after step (II) is d_p. But the variance that we want is $p(1-p)$, i.e. the variance of a population of infinite size generated on basis of p and no variance loss. So the factor of variance loss of step (II) is $\frac{d_p}{p(1-p)}$. Using for example the math program Maple this can be simplified to:

$$\frac{d_p}{p(1-p)} = \frac{1}{p(1-p)} \sum_{k=0}^{M} \frac{kM - k^2}{M^2} p^k (1-p)^{M-k} \binom{M}{k} = \mathbf{1 - \frac{1}{M}} \tag{8}$$

So we have:

- Factor of variance loss of step (II) :$1 - \frac{1}{M}$ (from (8)
- Factor of Variance loss of step (I) and (II) together :$1 - \frac{1}{N}$ (from [1])

Our factor y of variance loss of step (I) is therefor:

$$1 - \frac{1}{N} = y\frac{d_p}{p(1-p)} \Leftrightarrow y = \frac{1 - \frac{1}{N}}{\frac{d_p}{p(1-p)}} \tag{9}$$

With equation (8) we get $y = \frac{1-\frac{1}{N}}{1-\frac{1}{M}} = \frac{M(N-1)}{N(M-1)}$.

3.2 Distribution vector correction

If we use a different 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 is 1 (i.e. no variance loss) we will reduce the overall variance loss. For now we will try to reduce the variance loss of only step (I), the variance loss of step (II) will be discussed in chapter 4. As our variance loss in step (I) 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:

$$x = \frac{1}{y} = \frac{N(M-1)}{M(N-1)} \tag{10}$$

$$q(1-q) = p(1-p)x \Leftrightarrow -q^2+q-(-p^2+p)x = 0 \Leftrightarrow q_{1/2} = \frac{1}{2}(1 \pm \sqrt{1-4(-p^2+p)x}) \tag{11}$$

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:

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

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

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. In Figure 1 you can see the graph of the correction function for a population size of 10. The green line represents $f(p) = p$, i.e. no correction takes place. The larger the population size is the smaller is the horizontal line in the middle, i.e. the substitution with $p = 0.5$.

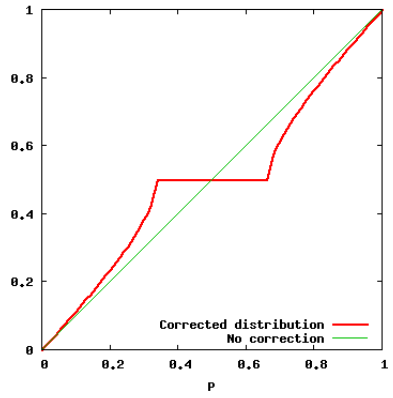

Fig. 1: Correction graph for Distribution Correction

3.3 Code

Finally we can put our results into a code, M is our population size N is our sampling size:

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

It is easy to implement the code piece within an existing application as it is problem independent (within the restrictions described above, bit-strings on a at 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 and before generating the new population.

4 Exact Distribution (ED)

So far we have calculated that the variance loss of generating a population on basis of p randomly (i.e. with RD) in step (II) will result in a variance loss of $1 - \frac{1}{M}$ (equation 8). 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 $1 - p$).
We can further reduce the variance loss if we create the new generation by distributing exactly $p \cdot M$ '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 will be no variance loss in step (II). You can see that in Figure 2, the methods using ED perform better in tness and in diversity. You can also see that it has a lower convergence rate with simple problems like OneMax which is because with Random Distribution the population can change faster.

For this method our p is calculated this way:

$$p = \frac{k_{old}}{N} \quad \text{and} \quad k_{new} = \lfloor pM \rfloor = \left\lfloor \frac{k_{old} M}{N} \right\rfloor$$

k_{old} denotes the number of '1's in the last generation, k_{new} denotes the number of '1's in the generation we are about to create. $\lfloor pM \rfloor = \lfloor \frac{k_{old} M}{N} \rfloor$ does only apply when we do not change p through any correction method (like Laplace Correction).

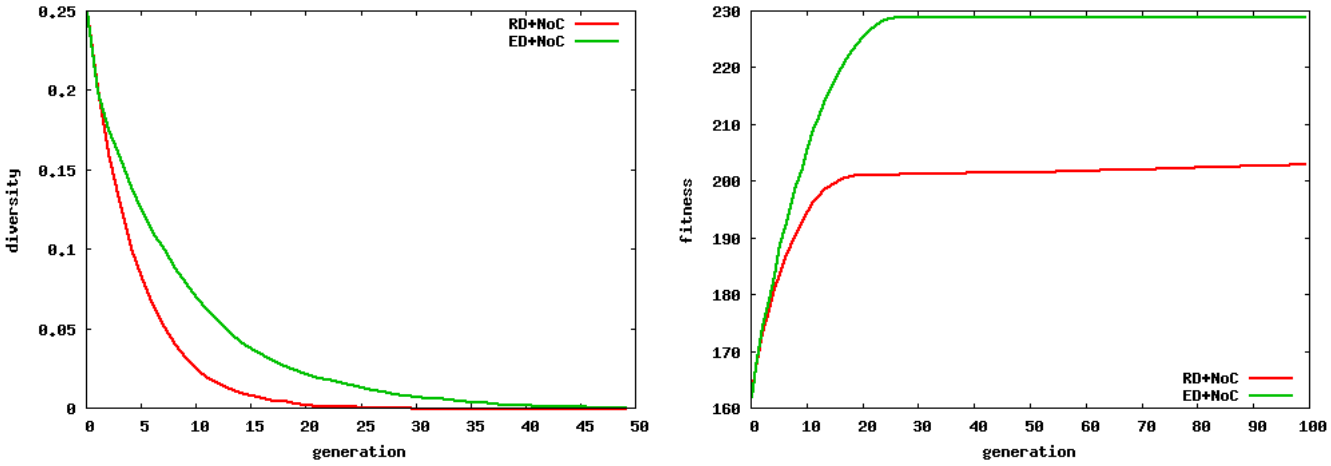

Fig. 2: Comparison between Exact Distribution and Random Distribution

4.1 Exact Distribution Correction (EDC)

When using an Exact Distribution the diversity loss in step (II) is expected to be di erent compared to the diversity loss of Random Distribution ($1 - \frac{1}{M}$). For now we will ignore the rounding error and assume that there are exactly $p \cdot M$ '1's in the population. We have to change in equation (5) the probability for $\tilde{p}_i = \frac{k_i}{M}$ accordingly:

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

The other equations (1), (2), (3), (4) about our variance v_i remain the same as they are dependent on our k_i anyways. Changes have to be made to equation (6) because our $P(\tilde{p}_i = \frac{k_i}{M})$ has changed. We no longer sum over all possible values of k as only $(k = p \cdot M)$ has the probability '1' while all other values for k have the probability '0':

$$\dot{d_p} = \sum_{i=1}^{C} v_{k_i = p_i M} P(\tilde{p}_i = \frac{k_i = p_i M}{M}) = \sum_{i=1}^{C} \frac{p_i M^2 - (p_i M)^2}{M^2} = \sum_{i=1}^{C} p_i(1 - p_i) \qquad (12)$$

As expected we have no loss of variance in step (II) using Exact Distribution because $\frac{\dot{d_p}}{p(1-p)} = 1$. From chapter 3 we know that the expected variance loss in step (I) is $\frac{M(N-1)}{N(M-1)}$ (equation 9) which is now also our expected total variance loss from step (I) and (II) together. Using same the correction method from chapter 3 our expected variance loss should therefor be zero instead of $-\frac{1}{M}$ as with the Random Distribution. In Figure 3 you can see that compared with RDC from the previous chapter 'Exact Distribution Correction' has the lowest variance loss on a fitness landscape.

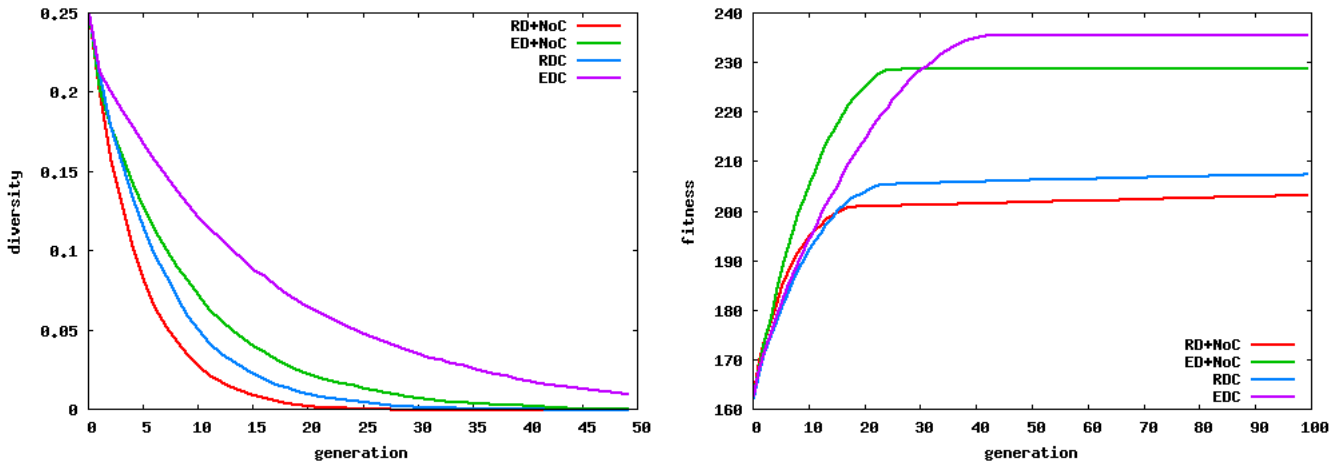

Fig. 3: Exact Distribution Correction has the lowest variance loss and reaches the highest fitness level

4.2 Rounding errors

The problem is that we actually cannot use the new distribution vector that we calculated with EDC to generate the new population because we have demanded that $p \cdot M$ is an integer.
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. But if we change the distribution vector then our new $p \cdot M$ will probably be no integer and we would have to change our EDC algorithm. So we have a mutual dependency between our algorithm and p.
Thus, when using Exact Distribution, we either solve that dependency which is difficult to accomplish (see chapter 4.3). Or we ignore the error in our algorithm and either round to the nearest integer or, depending on the remainder, randomly decide whether to add another '1' (e.g. for $p = 0.71$ and $M = 10$ we would distribute seven '1's and an eighth '1' with the probability of 0.1).
The drawback for the latter two methods is that the distribution no longer deserves the title 'exact' because we ignore the error from the ignored dependency although we still get a lower variance loss as seen in the tests. We will take a short look into the first method in the next section.

4.3 Real Exact Distribution Correction

As discussed in the previous section we cannot demand that our $p \cdot M$ is an integer while changing the p with EDC as discussed earlier. Using the second method mentioned in the Rounding Errors section (randomly decide whether to generate one additional '1') in equation (5) the probability for $\tilde{p}_i = \frac{k_i}{M}$ obviously changes as there are $\lfloor p \cdot M \rfloor$ '1's with the probability $1 - (p \cdot M - \lfloor p \cdot M \rfloor)$ and $\lfloor p \cdot M \rfloor + 1$ '1's with the probability $p \cdot M - \lfloor p \cdot M \rfloor$:

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

Again, the equations about our variance v_{k_i} remain the same and changes have to be made to equation (6) because our $P(\tilde{p}_i = \frac{k_i}{M})$ has changed and we no longer sum over all possible values of k but only two:

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

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

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

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

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

Our variance loss is as before $\frac{\dot{d}_p}{p(1-p)}$ which unfortunately cannot be simplified any further and therefore is dependent on p. With equation (9) we have

$$y = \frac{1 - \frac{1}{N}}{\frac{\dot{d}_p}{p(1-p)}} \quad \text{and} \quad x = \frac{1}{y} = \frac{\frac{\dot{d}_p}{p(1-p)}}{1 - \frac{1}{N}}$$

The problem is that we cannot easily insert that into equation (11) as in this case our x depends on the distribution vector p with which we create the new population. This is a problem because the creation of our new population in step (II) happens of course after the correction of p. So we really have to make x dependent on the distribution vector $\dot{q}$:

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

This is kind of difficult to solve for $\dot{q}$ and probably has to be solved numerically. On the one hand this 'corrected' correction would further increase the effectiveness of the erroneous 'Exact Distribution Correction' discussed earlier while on the other hand it would cost more calculation time. Additional tests have to be made to determine if it is worth it.

5 Problem types

5.1 Flat tness landscape

In this test we examine the behavior of the algorithms in terms of their diversity with varying parameters on a at tness landscape. It is basically 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 bit-string with length 10, i.e. we have 10 components.
In Figure 4 we can see the diversity loss of several di erent methods that were de ned earlier. The additional graph, '1 - 1/N loss', denotes the expected loss of diversity per generation according to [1] with Random Distribution, 'M(N-1)/(N(M-1)) loss' denotes the expected loss of diversity that we have shown in chapter 4 with Exact Distribution. You can see that a test run of both Random and Exact Distribution match approximately the loss that was expected.

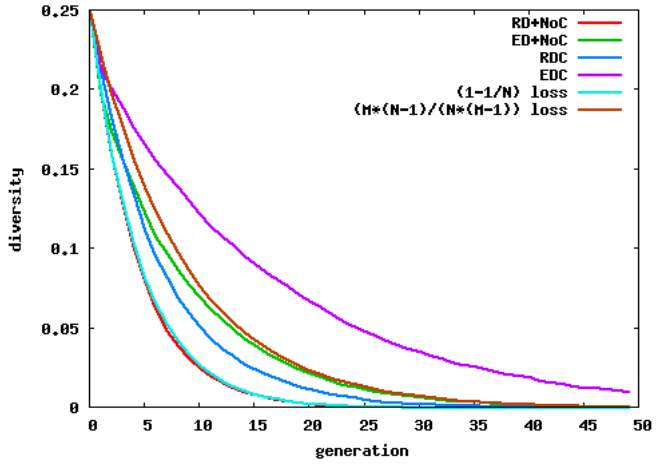

Fig. 4: Diversity loss on a at tness landscape (population size 10)

5.2 OneMax problem

In this test we examine the behaviour of the algorithms in terms of their fitness 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. This is also the main di erence to a at tness landscape, when selecting individuals for a new generation we do not randomly take N individuals and calculate our distribution vector p but we sort all individuals by their tness and only select the top 50%.

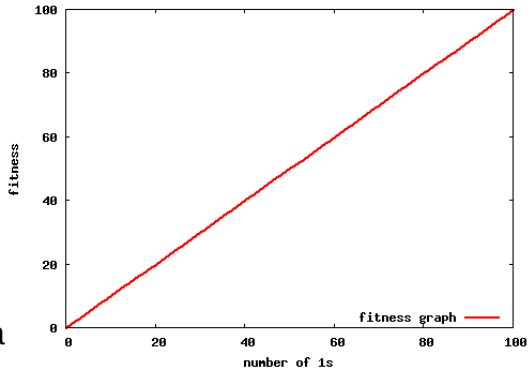

Fig. 5: Fitness Landscape of OneMax

While the OneMax problem does not represent a problem with a real at tness land-

scape (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 of the population are very similar (e.g. 010, 100, 001).
In our test the problem size ranges from 100 to 500 depending on the convergence speed of the functions that we will examine.

5.3 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 bit-position is important and the bits are connected, i.e. component i's tness depends on all components j with $0 < j < i$. This is more di cult to handle for UMDA's discussed in this report.
For this problem keeping correct solutions on lower bit-positions is very important, i.e. lower bit-positions contribute to the total tness more than higher bit-positions. Any algorithm that maintains a high diversity in positions that already have the correct value will have a signi cantly lower tness.

5.4 NK Landscape

From [3]:

An NK landscape is a real-valued function de ned on the set of binary n-tuples, $\{0,1\}^n$, which is of the form

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} f_i(x_i, \prod(x_i))$$

It is a summation of local tness functions f_i's, where each f_i depends on its main variable x_i and the variables in the neighborhood of x_i. Here the neighborhood $\prod(x_i)$ is a subset of the set $\{x_1, x_2, \ldots, x_n\} \backslash \{x_i\}$ and its size $|\prod_k(x_i)|$ is k. There are two ways to choose the variables in the neighborhood $\prod(x_i)$, adjacent neighborhood and random neighborhood. In the NK models with adjacent neighborhood, $\prod(x_i)$ consists of the closest k variables (with a certain to-break) to the main variable x_i with respect to the indices modulo n. In the NK models with random neighborhood, $\prod(x_i)$ is composed of the k variables chosen uniformly at random from $\{x_1, x_2, \ldots, x_n\} \backslash \{x_i\}$. Local tness functions are constructed independently of each other. For each local tness function, a random value from a probability distribution is assigned for each input.

We will use a similar implementation as [4], with 'Random(x)' denoting a function that maps x to a random value.

$$f_i = \frac{\text{Random}(2^k)}{2^k}$$

That means that two function calls with the same parameter (i.e. the same bit pattern) will result in the same value. If a new individual is created with the same pattern it gets

the same tness for this pattern. For $k = 1$ we would get a similar function as OneMax as there are only two possible values for f_k, i.e. we are counting '1's (or '0's). For large values of k this means that a large portion of memory is needed at the beginning (2^k e.g. 4 GB for $k = 32$) so for the implementation we will use a hashtable and relative small numbers ($k = 10$ and $n = 50$).

6 Additional methods

6.1 Boundary Correction (BC)

An optional correction method is called Boundary Correction . It can be used in connection with NoC, RDC and EDC and checks the boundaries of the nal value of the distribution vector p. In the case p gets above $1-\beta$ or below β it is corrected to these boundaries:

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

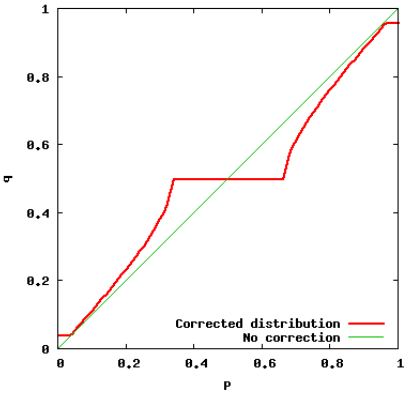

Fig. 6: Correction graph for Boundary Correction

6.2 Laplace Correction (LC)

Laplace correction is (so far) more or less the standard in connection with UMDA. It biases the resulting p towards $\frac{1}{2}$ and ensures that components do not get stuck at $p = 1.0$ or $p = 0.0$. Basically it behaves very much like Boundary Correction except that it cannot be combined with another correction method. The general formula for Laplace correction for a component i is to divide the number of '1's in the selected population plus α by the size of the selected population plus 2α.

$$p_i = \frac{k_i + \alpha}{N + 2\alpha} \tag{13}$$

Correction (LRC)

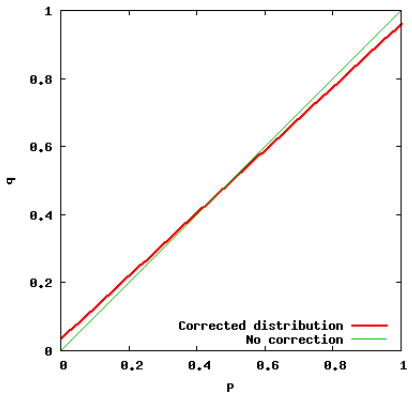

Fig. 7: Correction graph for Laplace

Another option is to include the distribution vector of the previous generation. We will replace α in the numerator with $\tilde{\alpha} = 2p_{t;i}\alpha$ where p_{t+1} denotes the new and p_t denotes the old distribution vector. In Figure 8 you can see that the function changes the new p to a lower value if the p_t was lower than 0.5. E ectively

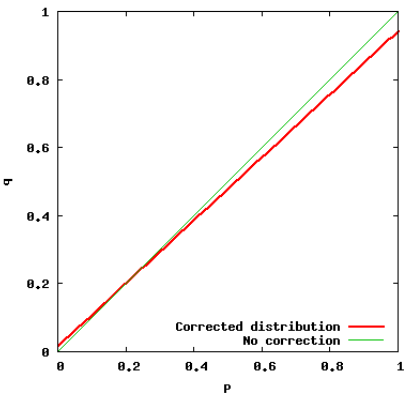

Fig. 8: Correction graph for Laplace Remember Correction with $p_t = 0.25$

this means that we soften sudden changes between two generations by a small degree.

$$p_{t+1;i} = \frac{k_i + 2\alpha p_{t;i}}{N + 2\alpha}$$

While including previous generations into the generation is not the scope of this report, it is certainly worth further examination. For example we can combine EDC with LRC by first executing EDC and then LRC. See chapter 8 for a more detailed analysis.

7 General test con guration

7.1 Graph drawing

The graphs were created with the help of gnuplot 4.0 and g++ 4.0 (gnu C++), the actual program is available on CD-ROM. 50 separate runs were made for each parameter setting and algorithm because all algorithms are based on random numbers. In addition each run consists of a number of generations. In each generation the best tness value that was found so far in that run is recorded. Then for each generations all recorded values are averaged between the runs and drawn as a graph. I.e. if we have one run (1 1 2 5 5 8) and another run (2 2 2 3 3 5) then our resulting graph would be (1.5 1.5 2 4 4 6.5).

7.2 Population size

Depending on the problem type, problem size and algorithm di erent population sizes ranging from 10 to 100 were tested. Also for each single con guration several di erent population sizes were tested because the convergence speed di er between the methods. Larger populations automatically result in more diversity and less diversity loss in each generation.

7.3 Selection size N

For problem types that do not use a at tness landscape we are interested in their tness. We usually select the top 50% from the population in order to calculate the distribution vector p. Higher selection rates will reduce the convergence speed because we are including below average solutions in p while lower selection rates will signi cantly increase the diversity loss. If the convergence speed is reduced the method will have dif- culties to reach or maintain good solutions either because the diversity dropped already to zero or (with Boundary Check or Laplace Correction) the high diversity holds back good solutions from spreading through the population. It is not the scope of this report to discuss this subject in detail but to present the general e ect of di erent selection sizes.

In Figure 9 we can see the trade-o between speed and quality and that elitism (i.e. small selection size) with methods that do not prevent uniformity in the population (i.e. no Boundary Correction or Laplace Correction) is very bad. With OneMax a selection size of 80% (or even higher) is the best choice because a high selection size simply maintains the diversity slightly longer above zero.

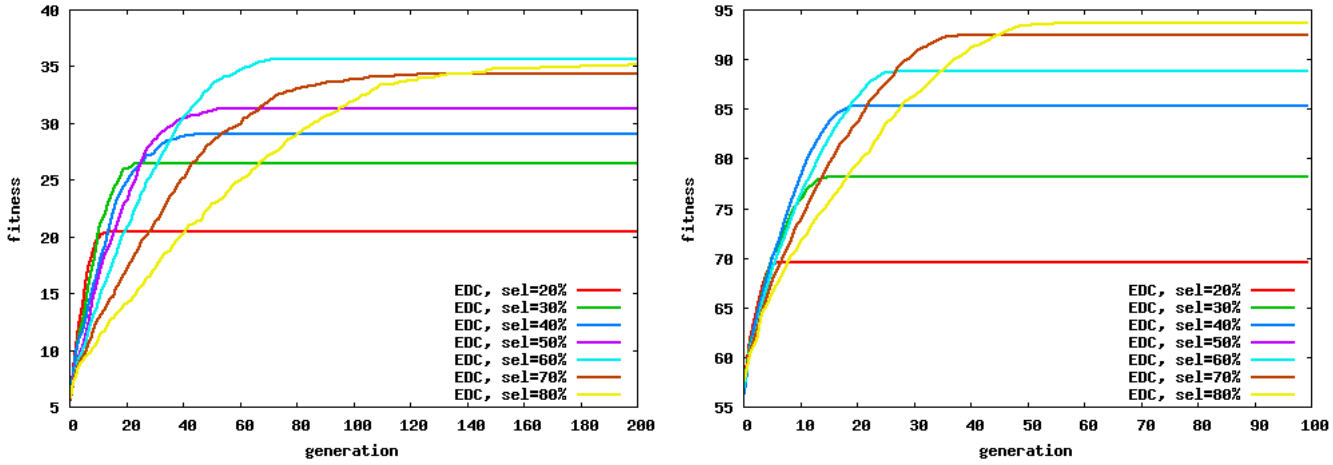

Fig. 9: Di erent selection sizes without boundary correction (LeadingOnes and OneMax, population size 50)

We can see this also with more di cult problems like the NK landscape (Figure 10), any extreme value for selection size generally does not improve the search.

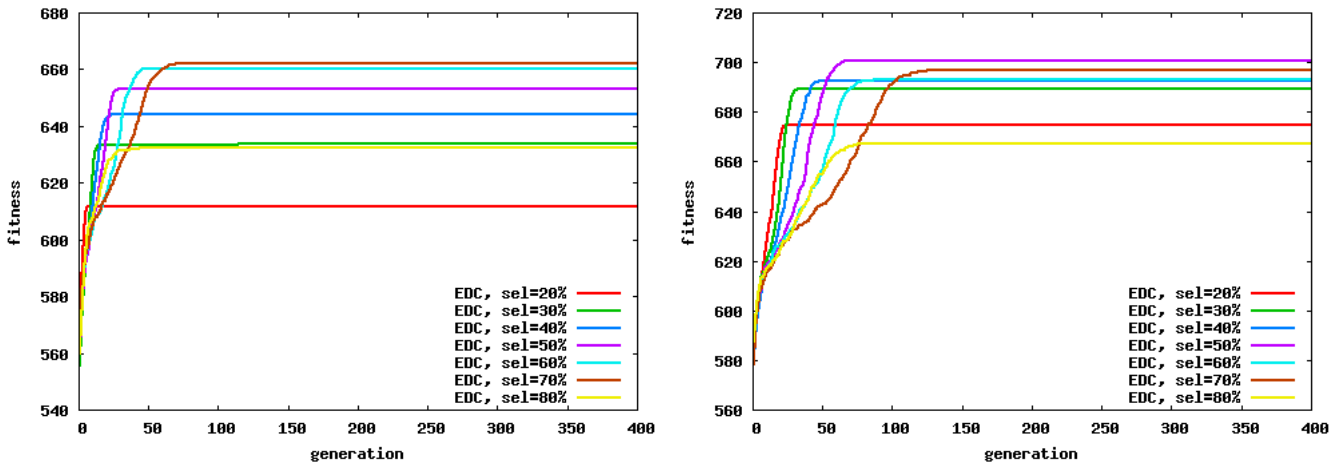

Fig. 10: Di erent selection sizes **without** boundary correction (NK-Landscape, population size 10 and 30)

On the other hand we can see in Figure 12 that elitism clearly pays o when using Boundary Correction with easier problems like LeadingOnes while being below average with more di cult problems like the NK landscape (Figure 11) or when using no Boundary Correction. So we can conclude that generally a selection size of 50% is a good standard value.

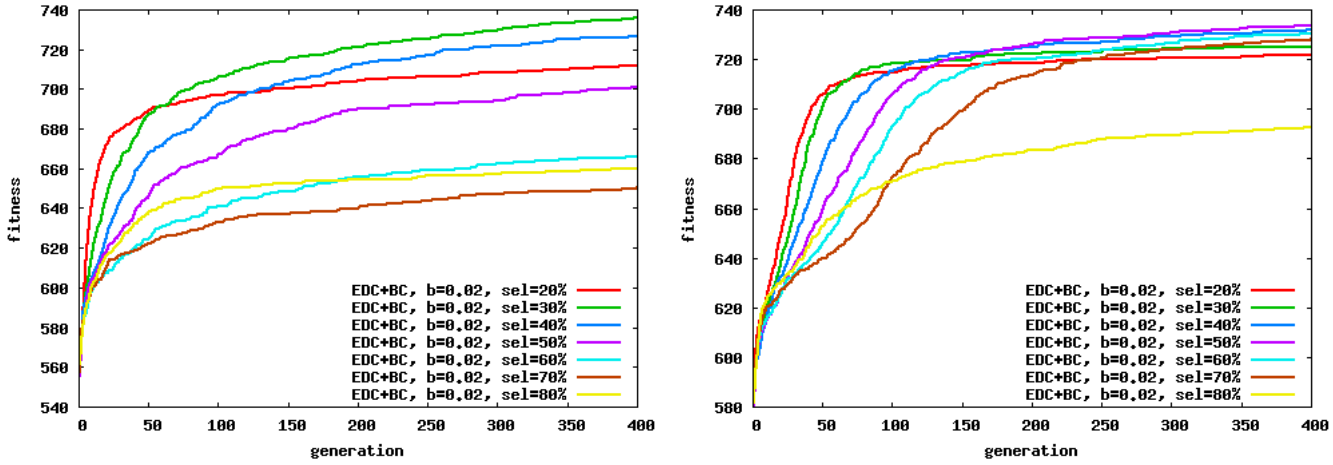

Fig. 11: Di erent selection sizes with boundary correction (NK-Landscape, population size 10 and 30)

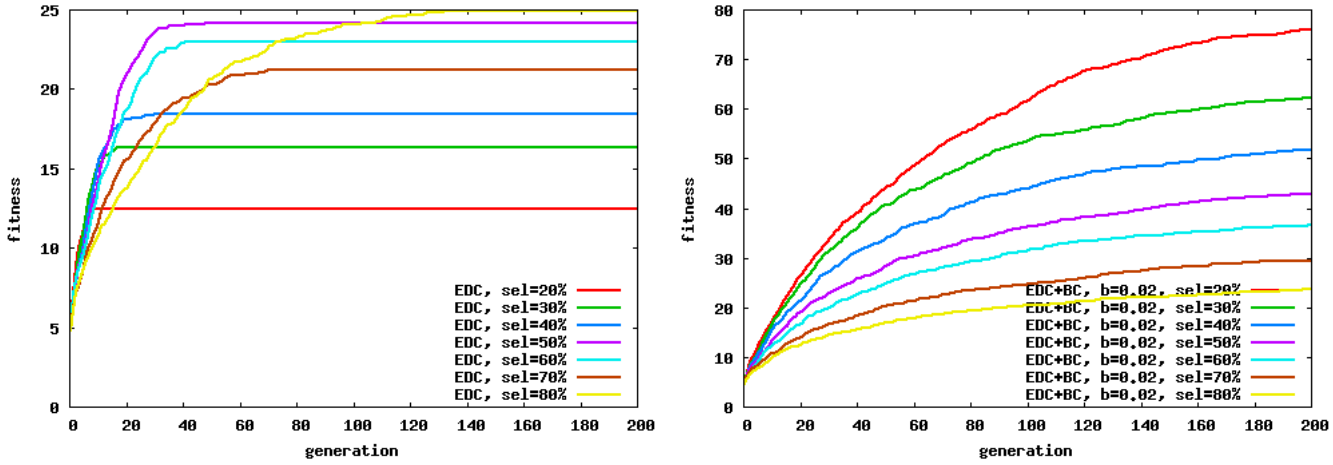

Fig. 12: Di erent selection sizes with and without boundary correction (LeadingOnes, population size 30)

8 Discussion and comparison of the correction methods

8.1 Boundary Correction

You can see in Figure 13 that with BC the diversity is held at a certain level. For each component there is always at least a chance of β to switch between '0' and '1'. On the one side this has the positive e ect that theoretically a better or even 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.

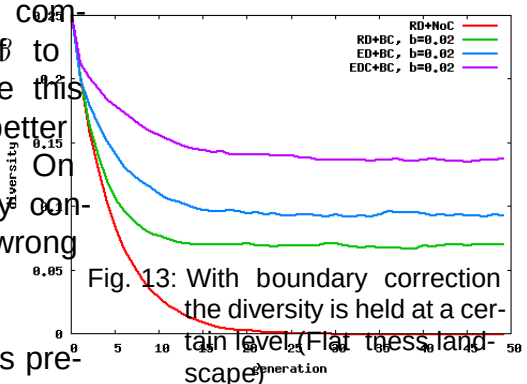

Fig. 13: With boundary correction the diversity is held at a certain level (Flat thess landscape)

If β is too large the function will have problems preserving the solution, if β is too low the function might need much more time to get out of a local optimum. For example in the case of an OneMax problem to get from 99 '1's to 100 '1's we not only need to generate another '1' but we have to preserve all the other 99 '1's, too. Thus we can conclude that a good value for β is dependent on all parameters, the population size, the problem type, the correction type and especially the problem size. In addition we can conclude that a very low value (relative to the problem size) for β will always be positive in the long run while reducing its convergence speed a little. In Figure 14 you can see that there values for β with which EDC+BC clearly is better than EDC without BC (i.e. $\beta = 0$) and in Figure 15 that the optimal value depends on the population size.

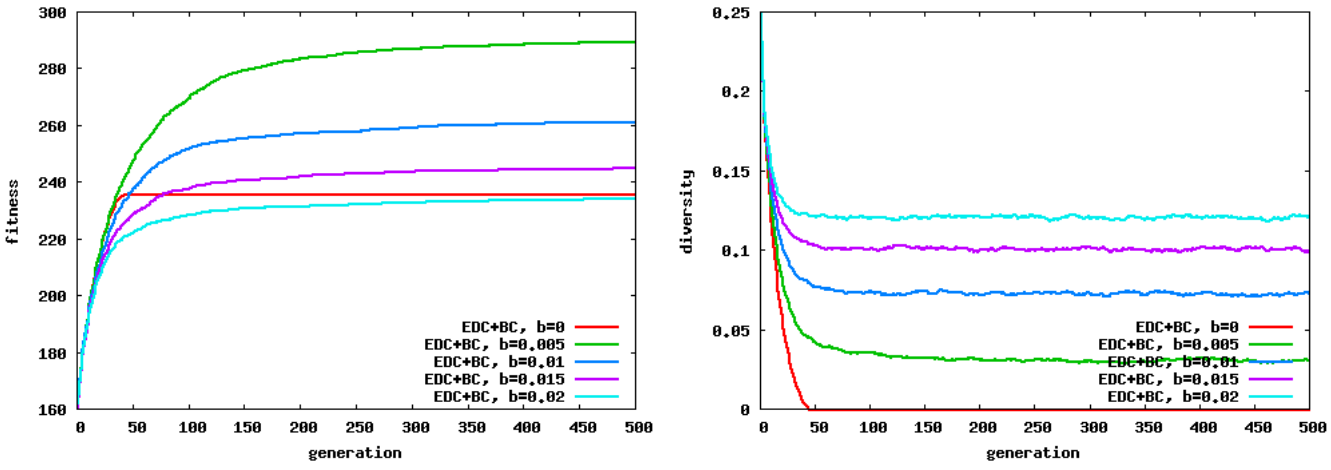

Fig. 14: Di erent values for β (OneMax, population size 10)

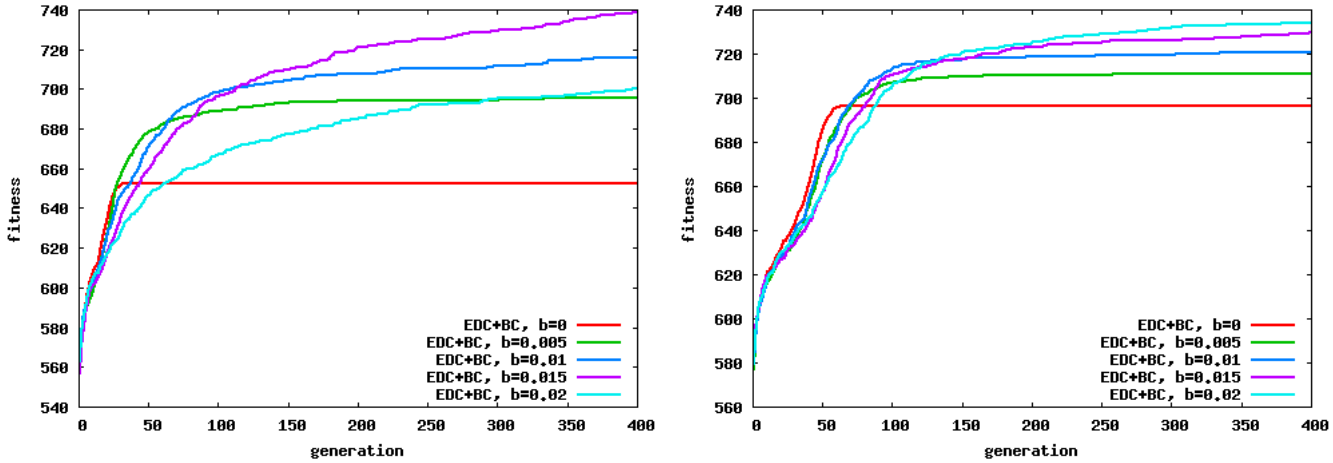

Fig. 15: Di erent values for β (NK-Landscape, population size 10 and 30)

Comparing EDC again with ED, this time with Boundary Correction, we see (Figure 16) that with simple problems like OneMax using EDC+BC will result in worse tness values. This is because EDC+BC maintains the diversity level at a higher level than ED+BC which makes it impossible to go beyond a certain number of '1's in the population.

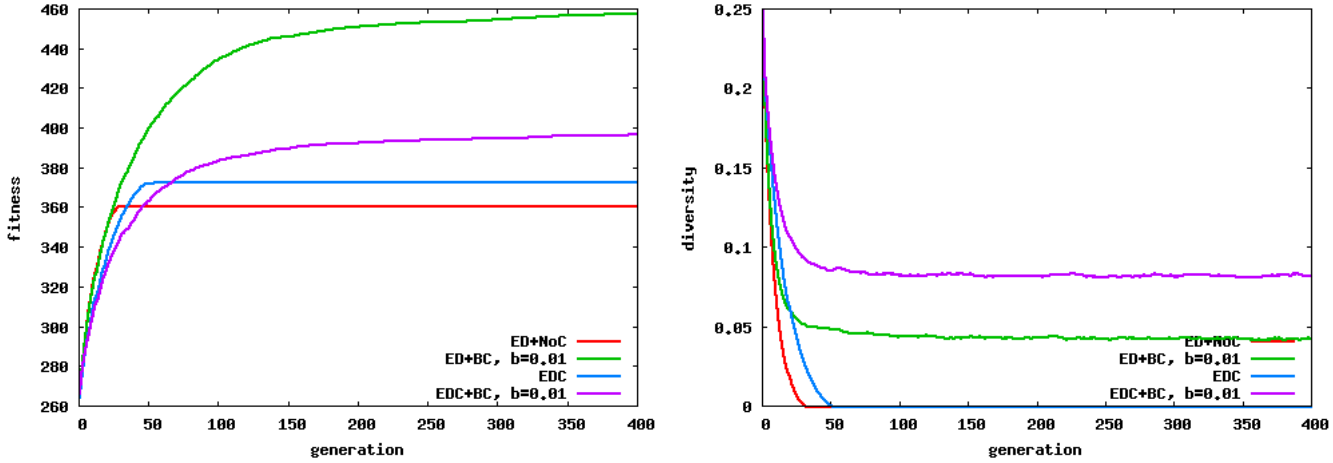

Fig. 16: The variance that EDC+BC maintains is too large in comparison with ED+BC (OneMax, population size 10)

When combining Boundary Correction with either RDC or EDC we will face the same problem as discussed with Exact Distribution. When we change our ψ before we create our population we will also have to adapt the Correction methods themselves. This could be a small source of error and probably have to be researched more closely if there is an algorithm that performs better than RDC+BC or EDC+BC by including β in its calculation.

8.2 Laplace Correction

Again the optimal value for α depends on the type of the problem and the problem parameters. As with Boundary Correction there can be a trade-o between convergence speed and tness for some problems as we can see in Figure 17. If we set α very low we have a chance to get out of uniformity of a component but a low convergence speed. If we use higher values for α we get faster to better solutions but are no longer able to keep good solutions.

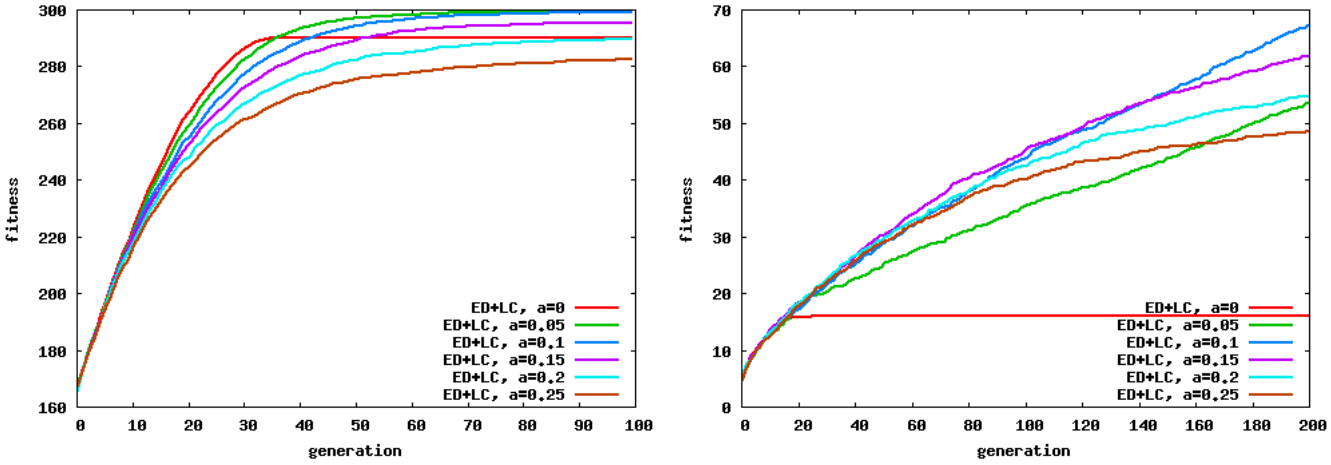

Fig. 17: α in uences the e ectiveness of Laplace correction but is problem-dependent, (OneMax and LeadingOnes, population size 30)

8.3 Boundary vs. Laplace Correction

As we have discussed earlier, both Laplace Correction and Boundary Correction do prevent p from getting near the boundary values $p = 0.0$ and $p = 1.0$. In addition LC biases the resulting p to $\frac{1}{2}$ due to the lower gradient. We can expect that Laplace Correction outperforms a simple Boundary Correction in terms of variance on a at tness landscape, the question is if this bias is signi cant.
For the test we will look at con gurations of α and β that change p at the borders to the same values. Laplace Correction corrects $p = 0$ to $\frac{\alpha}{N+2\alpha}$, Boundary Check corrects $p = 0$ to β, so in order to compare the behaviour at the borders we set $\beta = \frac{\alpha}{N+2\alpha}$. In Figure 18 we have set $\alpha = 0.1$ and $beta = 0.0193$ (for N = 5). As expected LC provides more variance but as we can see in the tness graph at the cost of a lower convergence speed.

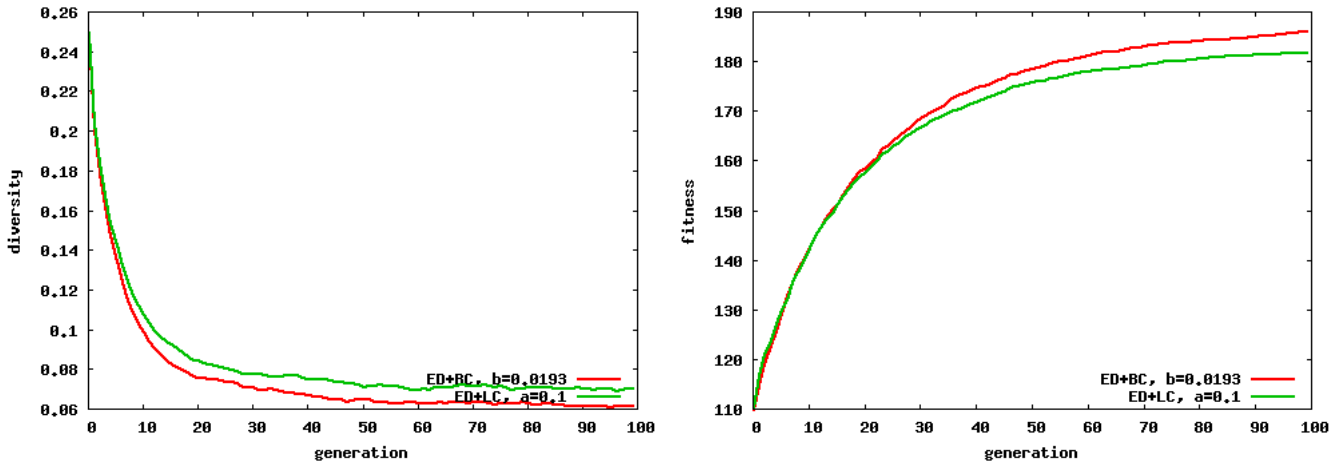

Fig. 18: Comparison of diversity and tness (OneMax, population size 10)

With a more di cult problem like a NK-landscape we see that LC again maintains a higher level of diversity which causes it to be slower at the beginning but reaching a higher level of tness later (Figure 19).

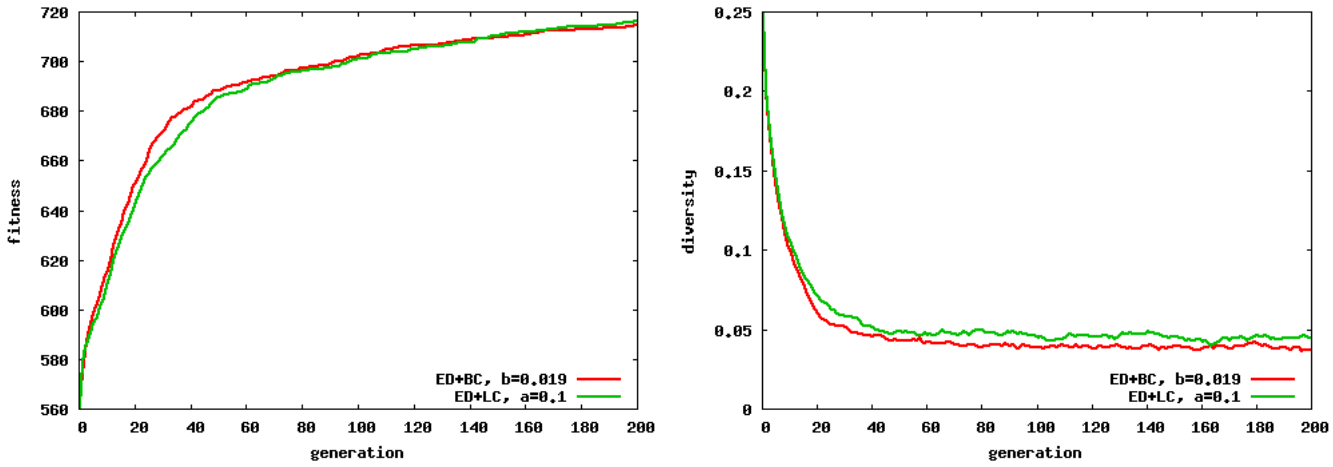

Fig. 19: Comparison of tness and diversity (NK-Landscape, population size 10)

In conclusion we can say that using Laplace Correction is not much di erent than Boundary Correction. Due to the lower gradient of the correction function it has a lower convergence rate which allows the method to search a longer in a population with higher diversity at the beginning. The main disadvantage of LC is that we can not easily combine it with other correction functions. Boundary Correction on the other hand only changes the borders of the correction function and will be used in this paper in connection with all other methods.

8.4 Laplace Remember Correction

Tests have shown that the value for α is not signi cant (see Figure 20) so we set $\alpha = 1.0$. As the method does not prevent extreme values for p we will also use a Boundary

Correction in connection with LRC.

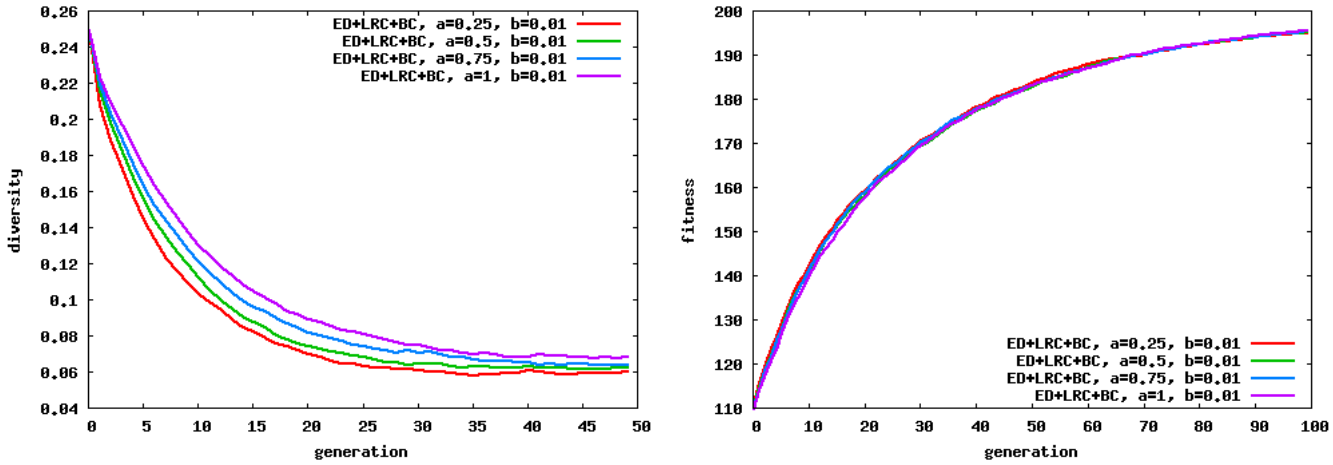

Fig. 20: No signi cant di erence between di erent values of α in diversity loss and tness (Flat tness landscape and OneMax, population size 10)

LRC behaves similarly as the underlying distribution function as can seen in Figure 21. Thus LRC alone does not give have any real advantage but we section we will try to combine LRC with EDC.

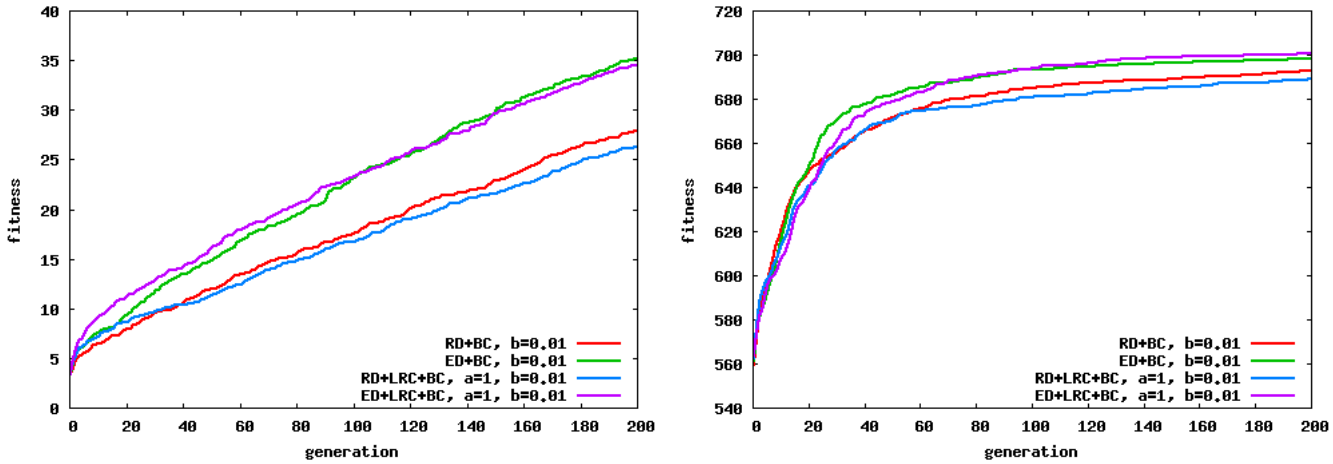

Fig. 21: Similar behavior of LRC as the underlying distribution method (Leading Ones and NK-Landscape, population size 10)

8.5 EDC with Laplace Remember Correction

As stated earlier we can combine EDC with LRC by rst executing EDC and then LRC with the parameters $p_{t;i}$ and $k_i = q_i \cdot N$, with q_i being the distribution vector corrected by EDC. Preventing sudden changes might be useful in connection with the Distribution Correction because the correction graph (Figure 1) has a large gradient left of p_1 and right of p_2.
In Figure 22 we can see that LRC does provide a higher level of diversity on a at tness

landscape and that it performs slightly better with the OneMax problem at the cost of a lower convergence speed.

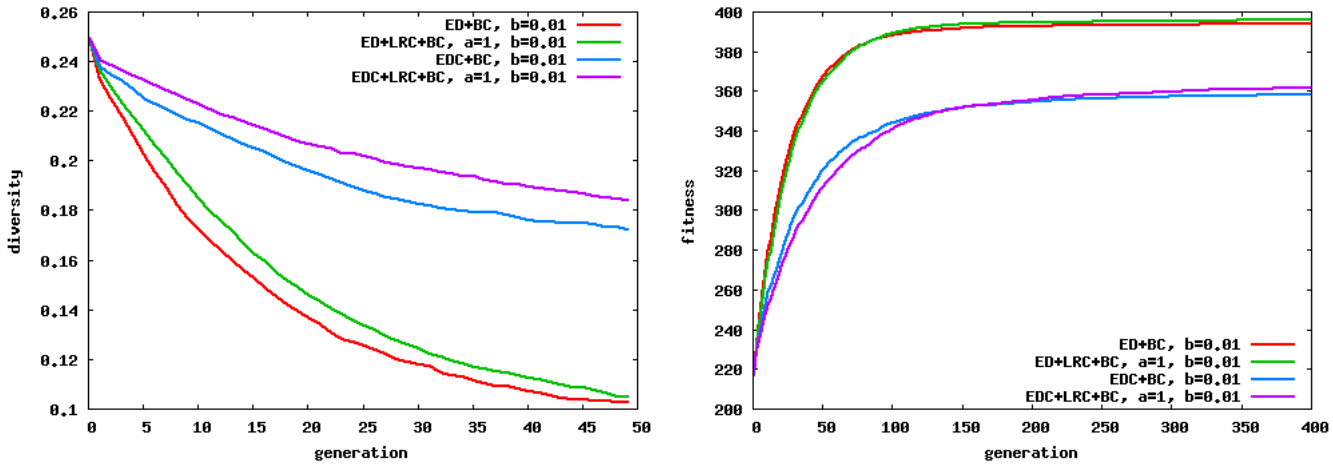

Fig. 22: Comparison between EDC+LRC and EDC in terms of their diversity and tness (Flat tness landscape and Onemax, population size 30)

While some runs show a better tness by a small margin, others do not (Figure 23. Thus it is problem dependent if LRC improves the method.

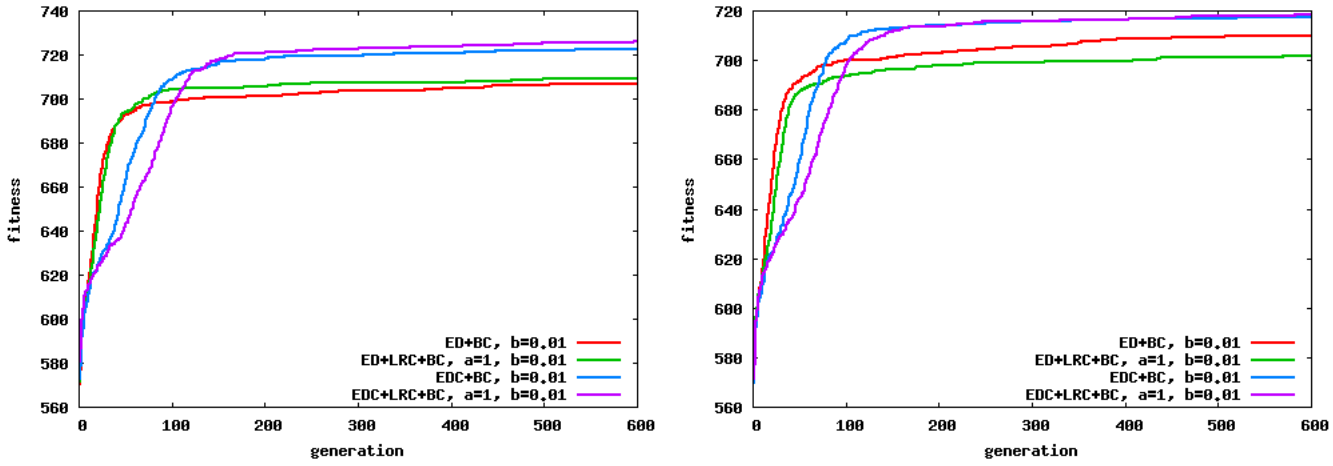

Fig. 23: LRC does not necessarily improve tness, only for some problem instances (NK landscape, same parameters, population size 20)

9 Summary

10 Fields of further research and summary

There are several open points in this report that were not discussed or left open for further research. Firstly the EDC can be improved by including the rounding error in the calculation. As the distribution vector p can hold only a limited number of values ($N + 1$ values) prior to the correction it is certainly possible to solve this numerically with a limited accuracy.
In addition the complete proof for equation 8 is missing and is only demonstrated by Maple to be correct.
Also there is missing a rule to determine the optimal values for β in the Boundary Correction method in connection with the parameters (problem size/type and population size). It was demonstrated that the results can be in uenced very negatively when choosing too large values for β.
Another point is that, if applied in a real application, the calculation time maybe another important factor. It was shown that EDC performed better than ED but only by a relatively small margin. EDC needs another calculation step and a table access respectively and needs more generations to reach the (higher) convergence level so it might be slower altogether.
Although it reaches sometimes a lower tness level, LRC in connection with EDC might be the answer concerning the convergence speed and tness trade-o . It needs more research to determine how to use the distribution vectors of previous generations e ectively.
At last other forms of EDAs where a dependency between the components is included in the calculation need to be researched using similar methods of correcting their diversity loss as demonstrated in this report with UMDAs.

References

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

[2] H. Muehlenbein and G. Paasz: From recombination of genes to the estimation of distributions: I. binary parameters, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, Parallel Problem Solving from Nature-PPSN IV, pages 178-187, Berlin, 1996. Springer.

[3] Y. Gao and J. Culberson: An Analysis of Phase Transition in NK Landscapes, http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume17/gao02a.ps.Z

[4] Potter, M.A.: NK-landscape problem generator, http://web.archive.org/web/20011120064244/cs.gmu.edu/~mpotter/nk-generator/ or http://www.cs.uwyo.edu/~wspears/nk.c