Theoretical Analysis of Simple Evolution Strategies in Quickly Changing Environments

Jürgen Branke[1] and Wei Wang[2]

[1] Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de
[2] Department of Education Technologies
Nanjing University of Posts and Telecommunications
P.O.Box 73, 38 GuangDong Road, 210003 Nanjing, China
wwang@jlonline.com

Abstract. Evolutionary algorithms applied to dynamic optimization problems has become a promising research area. So far, all papers in the area have assumed that the environment changes only *between* generations. In this paper, we take a first look at possibilities to handle a change during a generation. For that purpose, we derive an analytical model for a $(1, 2)$ evolution strategy and show that sometimes it is better to ignore the environmental change until the end of the generation, than to evaluate each individual with the most up-to-date fitness function.

1 Introduction

Many optimization problems are dynamic and change over time. A suitable optimization algorithm has to reflect these changes by repeatedly adapting the solution to the changed environment. Evolutionary algorithms (EAs) are inspired by natural evolution, which can be regarded as adaptation in an inherently dynamic and stochastic environment. Given this background, EAs seem to be naturally suited to be applied to dynamic optimization problems, and have already shown great promise (for an overview on the area, see e.g. [2]).

EAs are iterative algorithms. In each "generation", a number of new solutions (individuals) are generated, evaluated, and inserted into the population. So far, at least to the authors' knowledge, all publications on EAs for dynamic optimization problems assume that the environment (fitness function) changes *between* generations. Although this assumption is convenient, we consider it an oversimplification, because generally the environment is independent of the EA, and thus can change at any time, i.e. also *within* a generation.

In this paper, we specifically address the issue of how to handle an environmental change during a generation. For that purpose, we develop an analytical model for a $(1, 2)$ evolution strategy applied to the dynamic bit matching problem. The dynamic bit-matching problem is the dynamic variant of the well known onemax problem. The goal is to reproduce a binary target template, with the template changing over time. The fitness is just the number of bits identical with the template. We will analytically compare two ways to deal with a change of the target within a generation (i.e. after the first child has been evaluated): One possibility is to use the new fitness function for the second individual, while the other possibility would be to ignore the change and use the same old fitness function also for the second individual.

The first approach uses up-to-date fitness information, but potentially suffers from the fact that the selection is based on fitness from two different fitness functions. The second approach deliberately ignores new information about the enviroonment, but on the other hand, selection chooses between individuals evaluated with the same fitness function. Note that the second approach assumes that the old fitness function can still be used after the environment has changed. This seems justified, since in most practical applications, the fitness is evaluated by *simulating* the environment on a computer. Continuing to use the old fitness function then simply means to delay the update of the computer model.

The paper is structured as follows: in the next section, we will briefly mention a number of papers related to our work. In Section 3 we derive the theoretical model for the $(1,2)$ reproduction scheme and compare its performance to the $(1+1)$ reproduction scheme, assuming the environment changes *between* generations. Then, we turn to the issue of changes *within* a generation and compare the ideas of always using the up-to-date fitness function or using an old fitness function for the whole generation for the $(1,2)$ reproduction scheme. The paper concludes with a summary and an outlook for future work.

2 Related Work

Our paper is largely based on the work by Stanhope and Daida published in [5,6] but extends it significantly. Stanhope and Daida derive transition probabilities of the individual's fitness for a $(1+1)$ EA on the dynamic bit-matching problem. Let us briefly summarize their results, which will serve as a baseline for our extensions.

We will be using the following notation:

L	:	number of bits in the bit string. In our results reported below, we generally assume $L = 100$.
r	:	mutation rate in terms of the number of bits changed by mutation (fixed number, not probability per bit)
d	:	number of bits the target changes
a	:	parent individual
$f_t(b)$	:	fitness of individual b against the t-th target. The fitness after a change, i.e. against the next target, is usually denoted as $f_{t+1}(b)$. For reasons of readability, the subscript t is often omitted
$m_r(b)$	:	result of mutating individual b by r bits

The probability that an offspring has fitness x, given that it was generated by mutating an individual a with fitness $f(a)$, is the probability that the number of wrong bits mutated minus the number of correct bits mutated is equal to $x - f(a)$, which can be calculated as

$$P(f(m_r(a)) = x) = \frac{\binom{L-f(a)}{(x-f(a)+r)/2}\binom{f(a)}{r-(x-f(a)+r)/2}}{\binom{L}{r}} \tag{1}$$

In a $(1+1)$ EA, the better of the two individuals (parent and offspring) is kept, and the probability that the surviving individual has fitness x can thus be calculated as

$$P(\max_{b\in\{a,m_r(a)\}} f(b) = x) = \begin{cases} 0 & : \quad x < f(a) \\ \sum_{i=0}^{x} P(f(m_r(a)) = i) & : \quad x = f(a) \\ P(f(m_r(a)) = x) & : \quad x > f(a) \end{cases} \tag{2}$$

To account for changes of the environment, first note that a change of the target of d bits is equivalent (in terms of the individual's fitness distribution) to a d bit mutation of the individual. Thus we can calculate the distribution function of the fitness of a selected individual after an environmental change as:

$$P(f_{t+1}(\arg\max_{b\in\{a,m_r(a)\}} f(b)) = x) = \sum_{i=0}^{L} P(\max_{b\in\{a,m_r(a)\}} f(b) = i) P(f(m_d(c_i)) = x) \tag{3}$$

with f_{t+1} denoting the fitness against the new target (i.e. a target with d bits changed), and c_i denoting any individual with fitness i.

Further related work includes the paper by Droste [4]. There, the expected time to encounter the optimal solution for the first time is derived for a (1+1) EA on the dynamic bit matching problem. The model is different in that it assumes a mutation probability for each bit instead of a fixed number of bits inverted as we do. A brief survey on EA approaches to dynamic optimization problems has been presented e.g. in [1], a thorough treatment of different aspects of this subject can be found in [2].

3 Comparing $(1, 2)$ and $(1 + 1)$ on an Environment Changing *between* Generations

Let us now derive similar equations for the $(1, 2)$ reproduction scheme. Note that although now two new individuals are generated in every iteration, the total number of evaluations per generation is equal to the $(1+1)$ reproduction scheme when a change occurs after every generation, because the implicit assumption above was that the old individual is re-evaluated in every iteration in order to allow for correct selections.

Ignoring a change of the fitness function for now, the fitness distribution of the better of the two children can be described as:

$$P\left(\max_{b\in\{m_{r_1}(a),m_{r_2}(a)\}} f(b) = x\right) = P^2(f(m_r(a)) \leq x) - P^2(f(m_r(a)) < x) \tag{4}$$

which can be calculated using Equation 1.

The following equation takes a change after each generation into account:

$$P\left(f_{t+1}\left(\underset{b\in\{m_{r_1}(a),m_{r_2}(a)\}}{\arg\max} f(b)\right)=x\right)$$
$$=\sum_{i=0}^{L} P\left(\max_{b\in\{m_{r_1}(a),m_{r_2}(a)\}} f(b)=i\right) P(f(m_d(c_i))=x) \quad (5)$$

3.1 Optimal Mutation Rate

Figure 1 compares the optimal mutation rate (yielding the highest expected fitness of the selected individual) for $(1+1)$ and $(1,2)$ depending on the fitness of the current parent individual and assuming a string length of 100. The results are identical for a change severity of $d=0,1,2,3$. Obviously, for both approaches, with decreasing current fitness the optimal mutation rate increases quickly, as there is a higher chance for improvement and a lower chance for destroying valuable bits.

Since $(1+1)$ will never accept an individual worse than the current parent individual, it is safe to allow some mutation even when the parent has a high fitness. This is not the case for $(1,2)$, which risks a significant loss in fitness when mutation is introduced. For the examined case of a string length of 100, it is optimal to have a mutation rate of 0 as long as the parent's fitness is greater or equal to 73, and have a lower mutation rate than $(1+1)$ over the whole range.

It is also interesting to note that at least for $(1+1)$, mutating an even number of bits is never optimal. The reason is probably that an improvement can only occur when the child's fitness is actually *greater* than the parent's fitness. With an even number of bit-flips there is a relatively high probability that the effects of the different bit flips cancel out and the fitness of the individual is not changed at all (i.e. there can be no improvement). Using an odd number of bit flips "forces" mutation to change the individual's fitness. This leads to a larger number of actually better individuals, while the also larger number of worse individuals doesn't matter (since only better individuals are accepted).

3.2 Convergence Plots

The expected fitness distribution of next generation's parent individual corresponds to a transition matrix of a Markov chain. Assuming e.g. an initial random individual with fitness 50 and a total string length of 100, we can then compute the expected fitness of the parent individual over the generations. For the comparisons in this section, we used the optimal mutation rate for both approaches. Figure 2 shows the derived convergence plots for $d=0,1,3$.

As would be expected, in a stationary environment, $(1+1)$ clearly outperforms $(1,2)$. The onemax fitness function has no local optima and thus a local hill-climber such as $(1+1)$ works great. Our analysis here is restricted to the simple dynamic bit matching benchmark, but it would be interesting to compare

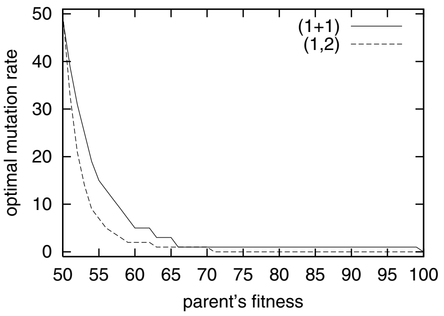
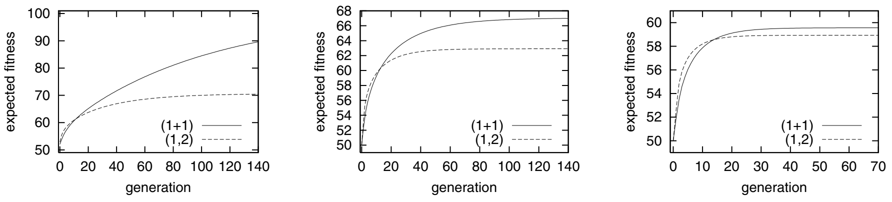


Fig. 1. Optimal mutation rate for $(1,2)$ and $(1+1)$ reproduction depending on the fitness of the current parent individual. Total string length is assumed to be 100.



(a) $d = 0$ (b) $d = 1$ (c) $d = 3$

Fig. 2. Comparison of the convergence curves for $(1+1)$ and $(1,2)$ for different change severities d. Total string length is assumed to be 100.

the two reproduction schemes also on a more rugged fitness landscape, where $(1+1)$ will get stuck in a local optimum.

As the environment starts to change, it is interesting to see that with increasing dynamism, the exploratory $(1,2)$ reproduction scheme comes closer and closer to the exploitatory $(1+1)$ reproduction scheme. In any case, $(1,2)$ outperforms $(1+1)$ while the parental fitness is still low, therefore it would be beneficial to use $(1,2)$ in the beginning of a run and then switch to $(1+1)$ later on.

4 Change *within* a Generation

In this section, we will consider the case where the fitness function changes during a generation of a $(1,2)$ reproduction scheme, i.e. after the first child has been evaluated. Given this simple framework, we will analytically compare the two strategies already mentioned in the introduction, namely to evaluate the two individuals with the respective (different) current fitness functions, or to artificially delay the change and use the old fitness function also for the second child.

As has been explained in the introduction, each of the above approaches has its advantages and its drawbacks, and we would like to know which approach is preferable given specific circumstances. Let us first consider the following illustrative example: Let x_1 and x_2 be the two children generated, and let $f(x_1) = 10$ and $f(x_2) = 12$ be the respective fitnesses *before* the environmental change, $f_{t+1}(x_1) = 11$ and $f_{t+1}(x_2) = 9$ be the respective fitnesses *after* the environmental change. The approach using the up-to-date fitness will select between $f(x_1) = 10$ and $f_{t+1}(x_2) = 9$, i.e. correctly select x_1 assuming a maximization problem. The approach delaying the change will compare $f(x_1) = 10$ and $f(x_2) = 12$ and select x_2, which is actually worse than x_1. On the other hand, in a situation where $f_{t+1}(x_1) = 8$ and $f_{t+1}(x_2) = 9$, the approach using up-to-date fitness is mistaken, while the approach which delays the change correctly selects x_2.

More generally, delaying the change will make the correct decision as long as the change does not affect the relative order of the two children. For the dynamic bit matching problem this is probably the case for very large bit strings and small mutation rates, because then x_1 and x_2 are identical for the vast majority of bits, and a change of the template will most likely affect them in the same way. In particular, if the environmental change is severe compared to mutation, the (undesirable) effect of comparing individuals with different fitness functions will "override" the fitness difference due to mutation, and mistakes are likely. In the following, we will compare the two strategies analytically.

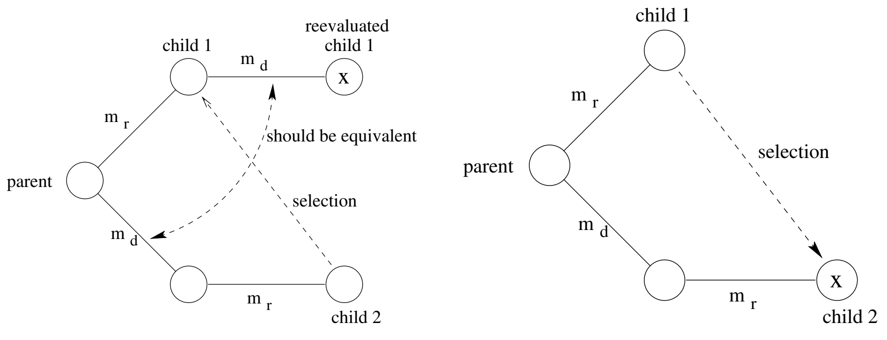
For the strategy which delays the change, the fitness distribution will be identical to the case considered in Section 3 with a change at the end of a generation. The situation for the other approach is depicted in Figure 3: First, a child is generated and evaluated, then the environment changes, a second child is generated and evaluated, and the child with the higher fitness is selected.

For analytical comparison, we need the *actual* fitness of the selected child, i.e. its fitness in the new environment. If the second child is selected, that is no problem, since it has already been evaluated against the new environment. However, if the first child is selected, the assigned fitness is outdated, and it has to be re-evaluated in the new environment (note that this re-evaluation is only necessary for the theoretical investigation and is not part of the implemented EA). The two cases are compared in Figure 3.

The difficulty is that the change before re-evaluating the first child has to be identical to the one which occurred before the evaluation of the second child. We have to be able to replicate that change, while at the same time we would like to continue using the high-level approach from the previous sections, avoiding to enumerate all possible changes on a bit-level. To solve this difficulty, let us first introduce the concept of two-step mutation.

4.1 Two-Step Mutation

Basically, a mutation of r bits can be regarded as first mutating $s < r$ bits, and then mutating the remaining $r - s$ bits on a shorter string, not containing the s bits mutated first (to avoid flipping the same bits twice).



(a) The first child is selected (b) The second child is selected

Fig. 3. Illustration of the process when the fitness function changes after the first child is evaluated, depending on whether (a) the first child is selected or (b) the second child is selected. We would like to derive the probability distribution for the fitness of the individual marked "x". Note that there is only one change of the environment, thus the two changes in (a) need to be identical.

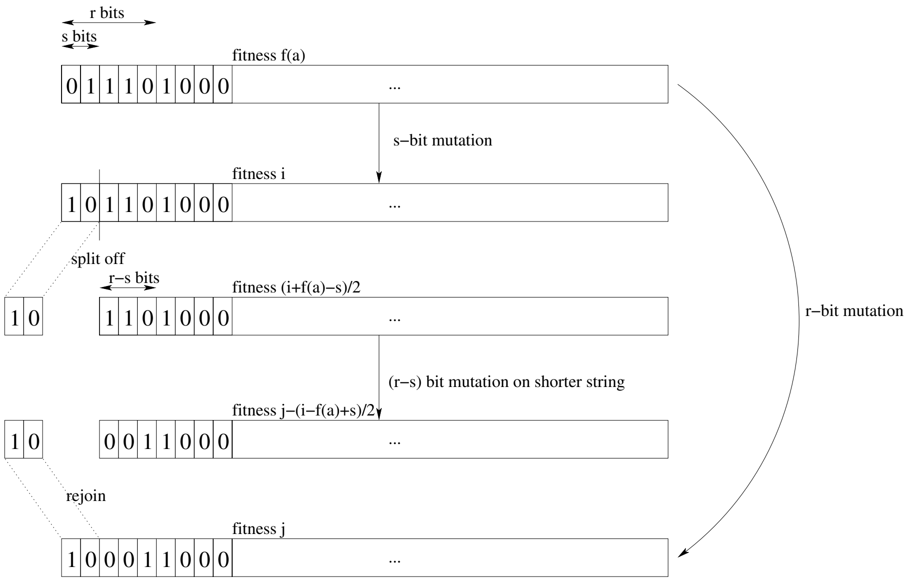


Fig. 4. Illustration of the concept of two-step mutation. Note that the order of the bits is irrelevant, thus w.l.o.g. we assume in this figure that the first bits are mutated.

The concept is illustrated in Figure 4: First, we do an ordinary s-bit mutation on the whole string. Let us assume that the resulting individual has fitness i. Then, the $r-s$-bit mutation can be captured by the basic mutation operation described by Equation 1, with the following parameters: The length of the substring is $L - s$, the mutation rate is $r - s$, the initial fitness of the substring is $(i + f(a) - s)/2$, and the fitness after mutation should be $j - (i - f(a) + s)/2$ (assuming the fitness of the individual after the whole two-step mutation is equal to j). We will denote such a mutation on a string of fitness $(i + f(a) - s)/2$ and length $L - s$ as $m_{r-s}(c^{L-s}_{(i+f(a)-s)/2})$. The probability distribution of the complete two-step mutation can then be expressed as

$$P(f(m_r(a)) = j) = \sum_{i=max\{0,f(a)-s\}}^{min\{L,f(a)+s\}} P(f(m_s(a)) = i) \cdot P\left(f(m_{r-s}(c^{L-s}_{(i+f(a)-s)/2})) = j - \frac{i - f(a) + s}{2}\right) \quad (6)$$

4.2 Using Two-Step Mutation to Model Change within a Generation

The above proposed two-step mutation can now be used to recover the change that has happened before the second child was evaluated, and apply it to the first child as well. The basic idea is to split the mutation into two steps: first, the s bits are mutated that are common to the r-bit mutation and the d-bit change of the target. Then, the remaining $r - s$ bits for mutation and the remaining $d - s$ bits to account for the environmental change are applied (see Figure 5 for illustration). Then, if we would like to apply the environmental change to the first child, we can do so by reversing the effect of the first s-bit mutation, and then applying a $d - s$ bit mutation on a string of length $L - r$ (since this last mutation has no common bits with the r-bit mutation).

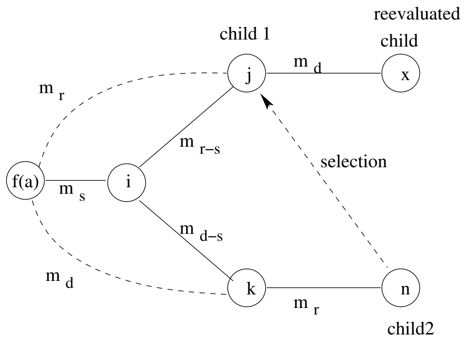


Fig. 5. Illustration of the use of two-step mutation for a (1,2) EA with a change of the environment after the first child has been evaluated, and assuming that the first child is selected. The letters inside the circles denote the fitnesses of the corresponding individuals.

The s-bit mutation and the $r-s$ bit mutation have already been discussed above. For the $d-s$ bit mutation, the string length is $L-r$ (since it has no common bits with either the s-bit mutation nor the $r-s$ bit mutation), the initial fitness is $(j+f(a)-r)/2$, and the fitness after mutation should be $k-i+(j+f(a)-r)/2$ (assuming the fitness of the individual after the whole two-step mutation is equal to k). Overall,

$$\begin{aligned}&P((f(m_d(a))=k)|(f(m_r(a))=j)\wedge(f(m_s(a))=i))\\&=P\left(f(m_{d-s}(c^{L-r}_{(j+f(a)-r)/2}))=k-i+\frac{j+f(a)-r}{2}\right)\end{aligned}\tag{7}$$

Let us first consider the case when the first child has an observed fitness equal or better than the second child, and let us assume that in this case, the first child is selected (selecting either child with equal probability in the case of equal fitness could also be handled, but is omitted here for clarity). As usual, we would like to calculate the probability that the child (in this case the first one) has fitness x.

The probability that the first child is selected can be calculated as

$$\begin{aligned}&\sum_{j=\max\{0,f(a)-r\}}^{\min\{L,f(a)+r\}}P(f(m_r(a))=j)P(f(m_r(m_d(a)))\leq j)\\=&\sum_{j=\max\{0,f(a)-r\}}^{\min\{L,f(a)+r\}}P(f(m_r(a)=j)\\&\cdot\sum_{k=\max\{0,f(a)-d\}}^{\min\{L,f(a)+d\}}P(f(m_d(a))=k)P(f(m_r(c_k))\leq j)\end{aligned}\tag{8}$$

Using two-step mutation, this can be re-formulated as

$$\begin{aligned}&\sum_{s=0}^{\min\{r,d\}}P(v(r,d)=s)\sum_{i=\max\{0,f(a)-s\}}^{\min\{L,f(a)+s\}}P(f(m_s(a))=i)\\&\cdot\sum_{j=\max\{0,i-r+s\}}^{\min\{L,i+r-s\}}P\left(f(m_{r-s}(c^{L-s}_{(i+f(a)-s)/2}))=j-\frac{i-f(a)+s}{2}\right)\\&\cdot\sum_{k=\max\{0,i-d+s\}}^{\min\{L,i+d-s\}}P\left(f(m_{d-s}(c^{L-r}_{(j+f(a)-r)/2}))=k-i+\frac{j+f(a)-r}{2}\right)\\&\cdot P(f(m_r(c_k))\leq j)\end{aligned}\tag{9}$$

where $P(v(r,d)=s)$ denotes the probability that a d-bit mutation and a r-bit mutation have exactly s common bits and can be calculated as

$$P(v(r,d)=s)=\frac{\binom{L}{s}\cdot\binom{L-s}{r-s}\cdot\binom{L-r}{d-s}}{\binom{L}{r}\cdot\binom{L}{d}}\tag{10}$$

The fitness of the first child after re-evaluation is determined by $x = f(a)+j+k-2i$. Since x is assumed to be known, k can be replaced by $k = x-f(a)+2i-j$.

The probability that the first child has been selected and has fitness x after re-evaluation can thus be calculated as

$$\sum_{s=0}^{\min\{r,d\}} P(v(r,d)=s) \sum_{i=\max\{0,f(a)-s\}}^{\min\{L,f(a)+s\}} P(f(m_s(a))=i)$$
$$\cdot \sum_{j=\max\{0,i-r+s\}}^{\min\{L,i+r-s\}} P\left(f(m_{r-s}(c^{L-s}_{(i+f(a)-s)/2}))=j-\frac{i-f(a)+s}{2}\right)$$
$$\cdot P\left(f(m_{d-s}(c^{L-r}_{(j+f(a)-r)/2}))=x+i-\frac{j+f(a)+r}{2}\right)$$
$$\cdot P(f(m_r(c_{x-f(a)+2i-j}))\leq j) \tag{11}$$

The second case, namely that the second child has a better observed fitness and is selected is much easier to handle. Since we don't have to re-evaluate the second individual, we just need to calculate the probability that it has fitness x while the first individual has fitness smaller than x. In terms of equations, this can be expressed as

$$\sum_{k=\max\{0,f(a)-d\}}^{\min\{L,f(a)+d\}} P(f(m_d(a)=k)P(f(m_r(c_k))=x)P(f(m_r(a))<x) \tag{12}$$

The total probability of the new parent having fitness x is then just the probability that the first child is selected and has actual fitness x (Equation 11) plus the probability that the second child is selected and has fitness x (Equation 12). As has been shown in [3], the presented framework can also be adapted to the general case of $(1,\lambda)$ evolution strategies with $\lambda > 2$.

4.3 Comparisons

The above equations allow us to compare the two strategies, namely to use the old fitness function for both individuals or to always use the latest fitness function, in terms of the expected fitness of the next generation's parent. Again, we assume a bit string of length 100 for the results reported below.

Figure 6 compares the difference in expected fitness of the next generation's parent individual depending on the current parent individual's fitness and the change severity d of the environment. As can be seen, for $d = 1$ it is always somewhat better to use the up-to-date information when evaluating individuals. This can also be derived differently: The fitness difference of the two children before the change can never be equal to 1, since in both mutations, an equal number of bits are flipped. If the fitness difference is 0, it is important to know the effect of the environment to correctly select the better individual. If the fitness difference ≥ 2, a change of a single bit can not reverse the ordering of the children and thus both fitness evaluation schemes will make the same decision.

However, for $d > 1$ such a simple analysis is not possible. Clearly according to Figure 6, with $d = 2$ or $d = 3$, using the old environment for the second child yields better results unless the parent's fitness is very low. As long as the parent's fitness is very low, the optimal mutation rate is very high, and the two children are likely to differ significantly in their true fitness. But with growing parental fitness and smaller optimal mutation rate, the children's true fitnesses may be quite similar, and when a severe change $(d > 1)$ is only taken into account for one individual, there is the danger that it will "hide" the true fitness difference, misleading selection. The "steps" in the lines in Figure 6 correspond to the points where the optimal mutation probability changes. The effect of different fitness evaluation schemes on the convergence curves can be seen in Figure 7.

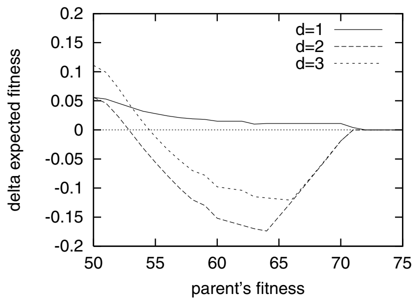


Fig. 6. Difference of expected fitness for next generation's parent individual depending on whether new or old fitness function is used for second child. If value is greater 0, it is better to use new fitness function and vice versa.

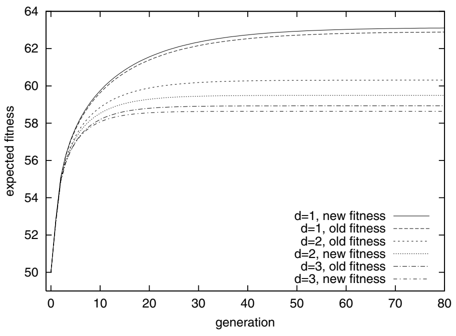


Fig. 7. Convergence curves of $(1, 2)$ reproduction strategy with either new or old fitness function used for the second child, for different change severities d. Lines are labeled in the order they appear in the plot.

5 Conclusion and Future Work

In this paper, we have approached the problem of quickly changing environments, and in particular the issue of how to handle a change of the fitness function that occurs *within* a generation of an evolutionary algorithm. Besides a general description of the issues and some ideas of how to handle such changes, we have derived analytic expressions which allow to calculate the probability distribution for fitnesses of the next generation's parent individual for the (1,2) EA on the dynamic bit matching problem. Using this framework, we compared two strategies to handle changes within a generation, namely to always use the up-to-date fitness information, or to keep the old fitness function despite the change of the environment. As has been shown, depending on the current fitness and the severity of the environmental change, one or the other strategy may be beneficial.

We are currently examining the issue of changes within a generation also from an empirical point of view, allowing us to look at much more complex problems and EA variants as suggested in the introduction.

Acknowledgments. We would like to thank Christopher Ronnewinkel for helpful discussions in the early phases of this research.

References

1. J. Branke. Evolutionary approaches to dynamic optimization problems - updated survey. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 27–30, 2001.
2. J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.
3. J. Branke and W. Wang. Theoretical analysis of simple evolution strategies in quickly changing environments. Technical Report 423, Institut AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany, 2002.
4. S. Droste. Analysis of the $(1+1)$ EA for a dynamically changing onemax-variant. In *Congress on Evolutionary Computation*, pages 55–60, 2002.
5. S. A. Stanhope and J. M. Daida. Optimal mutation and crossover rates for a genetic algorithm operating in a dynamic environment. In *Evolutionary Programming VII*, volume 1447 of *LNCS*, pages 693–702. Springer, 1998.
6. S. A. Stanhope and J. M. Daida. Genetic algorithm fitness dynamics in a changing environment. In *Congress on Evolutionary Computation*, volume 3, pages 1851–1858. IEEE, 1999.