Nat Comput (2010) 9:283–294
DOI 10.1007/s11047-008-9106-8

When to use bit-wise neutrality

Tobias Friedrich · Frank Neumann

Published online: 26 October 2008

Abstract Representation techniques are important issues when designing successful evolutionary algorithms. Within this field the use of neutrality plays an important role. We examine the use of bit-wise neutrality introduced by Poli and López (2007) from a theoretical point of view and show that this mechanism only enhances mutation-based evolutionary algorithms if not the same number of genotypic bits for each phenotypic bit is used. Using different numbers of genotypic bits for the bits in the phenome we point out by rigorous runtime analyses that it may reduce the optimization time significantly.

Keywords Evolutionary algorithms · Neutrality · Representations · Running time analysis · Theory

1 Introduction

Evolutionary algorithms (EAs) are randomized search heuristics that are inspired by the evolution process in nature. From biology it is known that many mutations in the genotype do not have any effect on the phenotype, i.e., they are neutral. This form of redundancy was first observed by Kimura (1968) when he tried to explain the high levels of polymorphism found within natural populations. The benefits of such neutral mutations have widely been discussed in the context of natural evolution (see e.g., Huynen 1996; Huynen et al. 1996; Schuster 2002). Such results from biology motivate the use of neutrality in evolutionary algorithms. Using neutrality in an evolutionary algorithm implies that additional redundancy is introduced into the considered search space. This research topic has attracted substantial interest in recent years. Several experimental studies have investigated

A preliminary version of this article appeared in *Proceedings of the IEEE Congress on Evolutionary Computation 2008.*

T. Friedrich · F. Neumann (✉)
Max-Planck-Institut für Informatik, Saarbrucken, Germany
e-mail: fne@mpi-inf.mpg.de

whether redundancy can significantly help to come up with better algorithms (Collins 2005; Rothlauf 2003; Toussaint and Igel 2003; Weicker and Weicker 2001).

We examine the use of neutrality from a theoretical point of view and take a closer look on bit-wise neutrality which has been introduced by Poli and López (2007). Bit-wise neutrality is perhaps the most simple and natural way to use neutrality when working with binary strings. In this model of neutrality the value of a phenotypic bit depends on a specific number of bits in the genome. The value of a phenotypic bit is determined by the corresponding genotypic bits and a chosen encoding function.

Our investigations point out that there is a direct correlation between the mutation probability in the genotype and the phenotype for the different encoding functions investigated by Poli and López (2007). Therefore working with this kind of neutrality in mutation-based evolutionary algorithms has only the effect of changing mutation probability. Due to this result it seems to be unnecessary to use bit-wise neutrality for such algorithms as the effect can also be obtained by changing the mutation probability directly in the phenotype.

Later on, we point out that the use of bit-wise neutrality is useful when considering different numbers of genotypic bits to encode the phenotypic bits. The reason for this is that the number of genotypic bits used for a phenotypic bit determines the mutation probability for this bit in the different encodings. We consider simple evolutionary algorithms and analyze the effect of bit-wise neutrality with different numbers of genotypic bits by carrying out rigorous runtime analyses. Analyzing the runtime time of evolutionary algorithms has become an important topic in the theoretical analysis of evolutionary algorithms (see e.g. Droste et al. 2002; He and Yao 2001) Using this kind of analysis, we point out that bit-wise neutrality can indeed help to speed up the computation of evolutionary algorithms. In particular, we examine plateau and deceptive functions and show that the proposed model of bit-wise neutrality can help to speed up the optimization process significantly if different numbers of genotypic bits are used to encode the bits in the phenome.

The outline of the paper is as follows. In Sect. 2, we introduce the model of bit-wise neutrality together with the different encodings we examine in the paper. Section 3 shows the correlation between the genotypic and phenotypic mutation rates. Optimal genotypic mutation rates are discussed in Sect. 4 and example functions where bit-wise neutrality using different numbers of genotypic bits is provably useful are presented in Sect. 5. Finally, we finish with some concluding remarks.

2 Model of neutrality

We are considering the search space $\{0,1\}^{\ell}$, i.e., each phenotype is a bitstring of length ℓ. We examine bitwise neutrality based on a genotype-phenotype mapping in the evolutionary process. In this form of neutrality each phenotypic bit is obtained from a group of genotypic bits via some encoding function. We consider three different kinds of genotype-phenotype encodings and assume the i-th phenotypic bit is encoded using a number of n_i genotypic bits. The encodings are defined as follows.

- **Parity encoding:** x_i is set to 1 if the number of ones among the n_i corresponding genotypic bits is even, otherwise x_i is set to 0.
- **Truth Table encoding:** A truth table is generated and the outcome is chosen randomly. 2^{n_i-1} randomly chosen assignments get output 0 and the other 2^{n_i-1} assignments get an output of 1. Considering n_i genotypic bits the phenotypic bit is chosen according to the corresponding output of the truth table.

- **Majority encoding:** x_i is set to 1 if the number of ones among the n_i corresponding genotypic bits is at least $n_i/2$, otherwise x_i is set to 0. We will only allow odd n to avoid draws.

In Poli and López (2007) these concepts of neutrality have been examined using the same number of n genotypic bits for each phenotypic bit, i.e., $n_i = n$ for all $1 \leq i \leq \ell$. In this case, one table is chosen that is used for each genotype-phenotype mapping in the Truth Table encoding.

Our aim is to examine the correspondence between the genotypic and phenotypic mutation rate in greater detail. Later on, we will examine in which situations it is useful to have different numbers of genotypic bits for the bits of the phenotype. This is motivated by neutrality observed in nature where different kind of information is encoded by parts of a DNA strand of different length.

3 Correspondence between phenotypic and genotypic mutation rates

We are interested in the relation between the genotypic mutation rate p_{ge} and the phenotypic mutation rate p_{ph} depending on the applied genotype-phenotype encoding. The understanding of this relation in important since the performance of an evolutionary process depends greatly on the right choice of the mutation rate. Poli and López (2007) already discovered that there is a direct correspondence between the genotypic and phenotypic mutation rate. In this section, we make this relation more comprehensible by deriving simple explicit equations mapping one to the other.

Parity encoding: For this encoding, Poli and López (2007) have pointed out that the mutation rate at phenotype level for the Parity encoding is given by

$$p_{ph} = \sum_{\substack{0 \leq i \leq n \\ i \equiv 1 (\mathrm{mod} 2)}} \binom{n}{i} p_{ge}^{i} (1 - p_{ge})^{n-1}.$$

In the following, we give a closed equation for this relationship that enables us to increase insight into the correspondence between the mutation rates in the genome and phenome.

$$\begin{aligned} p_{ph} &= \sum_{i=0}^{\lceil n/2 \rceil + 1} \binom{n}{2i+1} {p_{ge}}^{2i+1} (1 - p_{ge})^{n-2i-1} \\ &= \sum_{i=0}^{\lceil n/2 \rceil + 1} \sum_{j=0}^{n-2j-1} \binom{n-2j-1}{j} \binom{n}{2i+1} {p_{ge}}^{2i+1} (-p_{ge})^{j} \\ &= \sum_{i=1}^{n} \binom{n}{i} (-2)^{i-1} {p_{ge}}^{i} \\ &= \frac{1 - \sum_{i=0}^{n} \binom{n}{i} (-2\, p_{ge})^{i}}{2} \\ &= \frac{1 - (1 - 2 p_{ge})^{n}}{2}. \end{aligned} \tag{1}$$

To illustrate the correspondence between the mutation rates the function is shown in Fig. 1a and b for $n = 5$ and $n = 10$, respectively. Equation 1 and the two figures show that

there is a direct mapping between the genotypic and phenotypic mutation rate if the number of bits used in the genome is fixed. Note that $p_{ph}(p_{ge})$ is symmetric, i.e., $p_{ph}(p_{ge}) = p_{ph}(1 - p_{ge})$, for even n and antisymmetric, i.e., $p_{ph}(p_{ge}) = 1 - p_{ph}(1 - p_{ge})$, for odd n. The simple closed form of Eq. 1 allows us to derive the inverse function easily. That is, for odd n we get

$$p_{ge} = \begin{cases} \dfrac{1 - (1 - 2p_{ph})^{1/n}}{2} & \text{for } p_{ph} \leq 1/2 \\ \dfrac{1 + (2p_{ph} - 1)^{1/n}}{2} & \text{for } 1/2 \leq p_{ph} < 1. \end{cases} \tag{2}$$

As the mapping from p_{ge} to p_{ph} is not unique for even n, there are two inverse solutions for even n:

$$p_{ge} \in \left\{ \frac{1 - (1 - 2p_{ph})^{1/n}}{2}, \frac{1 + (1 - 2p_{ph})^{1/n}}{2} \right\} \tag{3}$$

Equations 2 and 3 are very useful when the optimal phenotypic mutation rate is known and we want to choose the corresponding genotypic mutation rate. Such an example is given in Sect. 4.

Truth Table encoding: When the Truth Table encoding is used, the phenotypic mutation rate is given by

$$p_{ph} = \frac{1 - (1 - p_{ge})^n}{2}. \tag{4}$$

For $n = 5$ and $n = 10$ this function is shown in Fig. 1c and d, respectively. Note that the phenotypic mutation rate is upper bounded by 1/2 independent of the genotypic mutation rate.

It is also interesting to observe that for $p_{ge} \leq 1/2$, the phenotypic mutation rate for the Truth Table encoding (cf. Eq. 4) is equal to the phenotypic mutation rate for the Parity encoding (cf. Eq. 1) if we half the genotypic mutation rate p_{ge}. Hence, both encodings result in the same phenotypic behavior if the Parity encoding uses half the mutation rate of the Truth Table encoding.

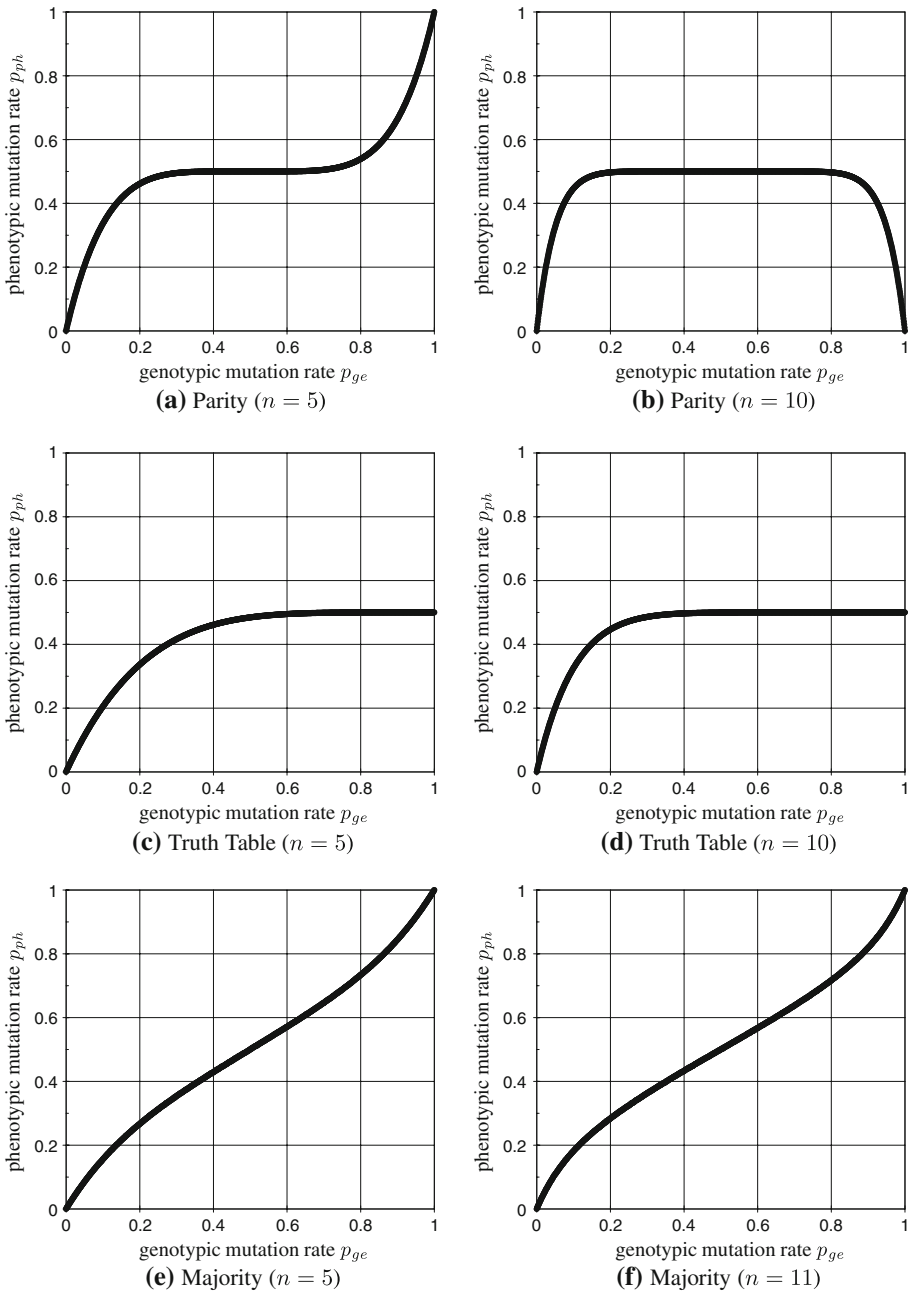
As Eq. 4 essentially describes the lower branch $p_{ge} \leq 1/2$ of Eq. 1, it easy to find its inverse function:

$$p_{ge} = 1 - (1 - 2p_{ph})^{1/n}. \tag{5}$$

Again, this can be used to obtain optimal genotypic mutation rates if the optimal phenotypic mutation rates are known.

Majority encoding: The Majority encoding is much harder to analyze as its effect on the phenotypic mutation rate depends on the current number of ones in the genotype. We can, however, obtain numerical estimates. To understand the mapping from the genotypic mutation rate to the phenotypic mutation rate, we have empirically examined the phenotypic effect of different genotypic mutation rates.

We approximate the resulting phenotypic mutation rate p_{ph} for a fixed genotypic mutation rate p_{ge} with the relative number of phenotypic changes for a sequence of 10^6 genotype mutations with mutation rate p_{ge}. That is, we start with a random genome and mutate each bit of the genome 10^6 times with mutation rate p_{ge}. Each time we count the number of zeros and ones in the genome. As the genome must have an odd number of genes, either the zeros or the ones hold the majority in the genome. We count the number



(a) Parity ($n = 5$) **(b)** Parity ($n = 10$)

(c) Truth Table ($n = 5$) **(d)** Truth Table ($n = 10$)

(e) Majority ($n = 5$) **(f)** Majority ($n = 11$)

Fig. 1 Mapping from genotypic mutation rate to phenotypic mutation rate for different encodings

the majority changes and set the resulting empiric phenotypic mutation rate to the number of majority changes divided by the number of runs (here 10^6).

The resulting functions are shown in Fig. 1e and f. The calculated mappings give a very good approximation of the dependence of the phenotypic mutation rate on the genotypic

mutation rate. This can be used to calculate the phenotypic mutation rate given the genotypic mutation rate and vice versa.

4 Optimal genotypic mutation rates

For many test functions the optimal phenotypic mutation rates are known. In this section, we derive the respective genotypic mutation rates for such cases.

For the ONEMAX-function on ℓ bits, it is well known that the optimization time is minimized at a phenotypic mutation rate of $p_{ph} = 1/\ell$ (see e.g. Droste et al. 1998). When the Parity encoding is used, the optimal genotypic mutation rate is therefore (for $\ell \geq 2$)

$$p_{ge} = \frac{1 - \left(\frac{\ell-2}{\ell}\right)^{1/n}}{2}.$$

Asymptotic in the problem size ℓ, this is

$$p_{ge} = \frac{1}{n\ell} + O\left(\frac{1}{\ell^2}\right).$$

Since p_{ge} doubles when using the Truth Table encoding instead of the Parity encoding, we get for the Truth Table encoding

$$\begin{aligned} p_{ge} &= 1 - \left(\frac{\ell-2}{\ell}\right)^{1/n} \\ &= \frac{2}{n\ell} + O\left(\frac{1}{\ell^2}\right). \end{aligned}$$

Poli and López (2007) have examined the runtime behavior of mutation-based EAs on ONEMAX depending on p_{ge} for $\ell = 14$ bits. With the above derived theory we can now calculate the optimal genotypic mutation rate for the ONEMAX problem. For $\ell = 14$ the optimal p_{ge} for the three different encodings and choice of n used by Poli and López (2007) are shown in Table 1. Using this table the experimental results given in Table 4 of Poli and López (2007) can be easily explained as it gets clear which genotypic mutation rate is close to the optimal mutation rate when considering the function ONEMAX.

Table 1 Optimal genotypic mutation rates for the ONEMAX-function on $\ell = 14$ bits

Encoding	Optimal p_{ge}
Parity ($n = 5$):	$p_{ge} \approx 0.0152$
Parity ($n = 6$):	$p_{ge} \approx 0.0127$
Parity ($n = 7$):	$p_{ge} \approx 0.0109$
Parity ($n = 8$):	$p_{ge} \approx 0.0095$
Majority ($n = 5$):	$p_{ge} \approx 0.0425$
Majority ($n = 7$):	$p_{ge} \approx 0.0377$
Truth Table ($n = 5$):	$p_{ge} \approx 0.0304$
Truth Table ($n = 6$):	$p_{ge} \approx 0.0254$
Truth Table ($n = 7$):	$p_{ge} \approx 0.0218$
Truth Table ($n = 8$):	$p_{ge} \approx 0.0191$

5 Benefits of bit-wise neutrality

In the following, we examine the case where the phenotypic bits may be encoded by a different number of genotypic bits. As pointed out in the previous sections, the mutation probability in the phenome depends on the genotypic mutation probability and the number of bits used to encode one phenotypic bit. Considering evolutionary algorithms, one usually works with a mutation probability that is the same for all bits. Hence, it seems to be natural to keep the genotypic mutation probability p_{ge} fixed and examine the effect of using different numbers of bits in the genome for the corresponding bits in the phenome.

We show that two popular evolutionary algorithms can only optimize certain functions in polynomial time if the phenotypic mutation rate is *not* fixed for all bits. We also prove that for fixed genotypic mutation rates this can be achieved by using different numbers of bits in the genome for each phenotypic bit. This shows a natural setting in which using neutrality improves the asymptotic runtime of an evolutionary algorithm.

An interesting example of different mutation rates in nature has been investigated by Stephens and Waelbroeck (1999). They observed that in the RNA sequences of the HI virus the mutability is lower in functionally important areas than in areas that tend to be recognized by a host's immune system. This is implemented by typically using different nucleotide triplets to encode the same amino acids in the two areas. In areas with high mutability the codons are likely to undergo non-synonymous mutations as only a few neighbors of the used codons are mapped to the same amino acid. On the other hand, in areas with low mutability codons with high neutral degree, that is, with a large fraction of neighboring codons that are mapped to the same amino acid, are used.

First, we investigate the function NH-ONEMAX defined by Gutjahr and Sebastiani (2008). It has been used for the analysis of evolutionary algorithms and ant colony optimization (Gutjahr and Sebastiani 2008; Neumann et al. 2007). The function is defined as

$$\text{NH} - \textsc{OneMax}(x) = \left(\prod_{i=1}^{k} x_i\right)\left(\sum_{i=k+1}^{n} x_i\right)$$

and consists of a NEEDLE-function on k bits and a ONEMAX-function on $n - k$ bits. The ONEMAX-part can only be optimized if the needle has been found beforehand. We call the first k bits the NEEDLE-part and the remaining $n - k$ bits the ONEMAX-part of a bitstring x. We consider the case $k = \ln n$ bits.

Gutjahr and Sebastiani considered the behavior of a simple evolutionary algorithm known as (1 + 1) EA* in the literature (Jansen and Wegener 2001) on this function. The algorithm can be defined as follows.

Algorithm 1 (1 + 1) EA*

Choose an initial solution $x \in \{0,1\}^n$ uniformly at random.
repeat
  Create x' by flipping each bit of x with probability p_{ph}.
  if $f(x') > f(x)$ **then** set $x := x'$.
until stop

The optimization time of an evolutionary algorithm is defined as the number of fitness evaluations until an optimal search point has been obtained for the first time. Often the expectation of this value is considered and called the expected optimization time.

Gutjahr and Sebastiani showed a superpolynomial lower bound on the expected optimization time of the (1 + 1) EA* on NH-OneMax when the standard choice $p_{ph} = 1/\ell$ is used. We generalize this result and show a superpolynomial lower bound that holds for each fixed choice of p_{ph}.

Theorem 1 *The optimization time of the (1 + 1) EA* for each fixed choice of p_{ph} on* NH-OneMax *is superpolynomial with probability 1 − o(1).*

Proof We distinguish two cases and show that for $p_{ph} \leq n^{-1/2}$ the (1 + 1) EA is not able to optimize the Needle-part while for $p_{ph} \geq n^{-1/2}$ the onemax-part can not be optimize.

We consider the case $p_{ph} \leq n^{-1/2}$ first. The initial solution has at most $k - (\ln n)/3$ ones in the Needle-part with probability $1 - o(1)$ due to Chernoff bounds. As long as the needle has been found steps no other solutions is accepted. The probability to produce from a solution with at most $k - (\ln n)/3$ ones in the Needle-part the needle is upper bounded by $(\frac{1}{\sqrt{n}})^{(\ln n)/3} = n^{-(\ln n)/6}$ which implies that the optimization is superpolynomial with probability $1 - o(1)$ in this case.

For the case $p_{ph} \geq n^{-1/2}$ holds we consider the onemax-part. Let $r = \sum_{i=k+1}^{n} x_i$ be the number of ones in the onemax-part of the current solution x. For the initial solution $n/3 < r < (2/3)n$ holds with probability $1 - e^{-\Omega(n)}$ using Chernoff bounds. For the next accepted solution the needle has to be found as otherwise no improvement can be achieved. The expected number of ones that are turned into zeros in the onemax-part is $r \cdot p_{ph}$ and the expected number of zeros turned into ones is $(n - r) \cdot p_{ph}$. This implies that the number of ones that are turned into zeros is with probability $1 - e^{-\Omega(\sqrt{n})}$ at least $\Omega(\sqrt{n})$ using Chernoff bounds once more. Hence, an optimal solution has not been achieved with probability exponentially close to 1 when the needle has been found for the first time.

We consider the point of time where $r \geq (3/4) \cdot n$ holds for the first time. Note, that an optimal solution has not been reached at this time as $\Omega(\sqrt{n})$ 1-bits flip with $1 - e^{-\Omega(\sqrt{n})}$ in a step that leads to this situation. After having achieved $r \geq (3/4) \cdot n$, expected number of ones turned into zeros is at least $(3/4)\ p_{ph}n$ and at least $(2/3)\ p_{ph}n$ with probability $1 - e^{-\Omega(\sqrt{n})}$ using Chernoff bounds. Similarly, the expected number of zeros turned into ones is at most $(1/4)\ p_{ph}\ n$ and most $(1/3)\ p_{ph}\ n$ with probability $1 - e^{-\Omega(\sqrt{n})}$ using Chernoff bounds. Therefore, the number of ones in the onemax-part decreases by at least $(1/3)p_{ph}n \geq \sqrt{n}/3$ with probability $1 - e^{-\Omega(\sqrt{n})}$ which implies that the number of steps needed to increase the number of ones in the onemax-part is exponential with probability exponentially close 1 after having reaching a search point that has at least $(3/4) \cdot n$ ones in onemax-part. □

In the following, we point out how bit-wise neutrality using different number of genotypic bits for the phenotypic bits may help to reduce the runtime of the (1 + 1) EA* significantly.

We investigate a model of bit-wise neutrality using the parity encoding although the result can also be shown for other models of bit-wise neutrality. The mutation rate is $p_{ge} = 1/\ell$ for each genotypic bit but different numbers of genotypic bits for the bits in the phenome are used. We choose $n_i = 2\ell$ for $1 \leq i \leq k$ and $n_i = 1$ for $k + 1 \leq i \leq \ell$. Hence, the number of bits in the genome is $2\ell k + \ell - k$ and we apply the evolutionary algorithm to the search space

$\{0,1\}^{2\ell\ k+\ell-k}$. Note, that the fitness evaluation still takes place on the basis of the corresponding phenotypic bits, i.e., a genotype is decoded before fitness evaluation.

The resulting mutation probabilities for the bits in the phenome can be computed using Eq. 1. It holds

$$p_{ph}(x_i) = \frac{1-(1-2/\ell)^{2\ell}}{2} \geq 1/e, \quad 1 \leq i \leq k$$

and

$$p_{ph}(x_i) = \frac{1-(1-2/\ell)^{1}}{2} = 1/\ell, \quad k+1 \leq i \leq \ell.$$

Using this setting we can prove that the runtime behavior of the (1 + 1) EA* changes significantly. In particular the expected optimization time on NH-OneMax becomes a polynomial of small degree.

Theorem 2 *Using the (1 + 1) EA* with $p_{ge} = 1/\ell$ together with the parity encoding where for each x_i, $1 \leq i \leq k$, of the phenotype 2ℓ genotypic bits and for each x_j, $k+1 \leq j \leq n$, of the phenotype 1 genotypic bit is used, the expected optimization time on* NH-OneMax *is $O(n^2 \log n)$.*

Proof Each bit on the NEEDLE-part in the phenotype is flipped with probability at least $1/e$. The probability that a specific bit in the phenotype is not flipped is at least 1/2. Hence, a solution x with k leading ones is produced with probability at least $(1/e)^{\ln}n$ in the next step. This means that the expected number of steps to produce a search point consisting of k leading ones is $O(n)$ and holds independently of the current solution. Each solution with k leading ones that has at least one 1-bit in the ONEMAX-part is accepted. Assuming that all bits in the ONEMAX-part are zeros the expected waiting time to flip one of these bits is $O(\frac{n-k}{n}) = O(1)$. Hence, the expected time to produce an accepted solution where the needle is found and the number of ones in the ONEMAX-part is at least 1 is $O(n)$. After this the needle will not be lost and the number of ones in the ONEMAX-part can only increase until an optimal solution has been found.

The (1 + 1) EA* with mutation rate $1/\ell$ optimizes the function ONEMAX in an expected number of $O(n \log n)$ steps (Droste et al. 2002). As the needle is re-sampled after an expected number of $O(n)$ steps the $O(n^2 \log n)$ bound on the expected optimization time follows. □

Often EAs replace equally good search points in the selection steps. In this case, they are able to deal with plateaus of moderate size. The following algorithm called (1 + 1) EA uses this selection methods and is frequently used for the runtime analysis.

Algorithm 2 (1 + 1) EA

Choose an initial solution $x \in \{0,1\}^n$ uniformly at random.
repeat
  Create x' by flipping each bit of x with probability p_{ph}.
  if $f(x') \geq f(x)$ **then** set $x := x'$.
until stop

It is not to hard to show that the (1 + 1) EA with $p_{ph} = 1/\ell$ optimizes the function NH-OneMax in expected polynomial time by using results of the optimization of the 1 + 1 EA

on NEEDLE (see e.g. Garnier et al. 1999; Wegener and Witt 2005). However, this algorithm has difficulties when replacing the NEEDLE-part by a TRAP-part that makes the problem deceptive.

The function TRAP-ONEMAX differs from NH-ONEMAX by the role of the first k bits. It is defined as

$$\text{TRAP-ONEMAX}(x) = \left(\prod_{i=1}^{k} x_i\right)\left(\sum_{i=k+1}^{n} x_i\right) + \sum_{i=1}^{k}(1 - x_i).$$

Similar to NH-ONEMAX, we call the first k bits the TRAP-part and the remaining $n - k$ bits the ONEMAX-part of a bitstring x and consider the case $k = \ln n$. We first investigate the case where each phenotypic bit has the same mutation rate p_{ph} and show that the $(1 + 1)$ EA is not efficient on TRAP-ONEMAX in this case.

Theorem 3 *The optimization time of the (1 + 1) EA for each fixed choice of p_{ph} on* TRAP-ONEMAX *is superpolynomial with probability $1 - o(1)$.*

Proof Again, we distinguish two cases and show that for $p_{ph} \leq n^{-1/2}$ the $(1 + 1)$ EA is not able to optimize the TRAP-part while for $p_{ph} \geq n^{-1/2}$ the ONEMAX-part cannot be optimize.

We consider the case $p_{ph} \leq n^{-1/2}$ first. The initial solution has at most $k - (\ln n)/3$ in the TRAP-part with probability $1 - o(1)$ due to Chernoff bounds. As long as no solution with k leading ones has been found, steps that increase the number of ones in the TRAP-part are not accepted. Hence, the probability to produce a solution with k leading ones is upper bounded by $(\frac{1}{\sqrt{n}})^{(\ln n)/3} = n^{-(\ln n)/6}$ which implies that the optimization time is superpolynomial with probability $1 - o(1)$ in this case.

For the case $p_{ph} \geq n^{-1/2}$ holds we consider the ONEMAX-part. Thereby, we neglect the time needed to reach the optimum on the TRAP-part. Note that as long as the optimum has not been found on the TRAP-part the optimization process is completely independent of the ONEMAX-part. As each bit is flipped with the same probability, we may assume that the bits on the ONEMAX-part are uniformly distributed when the optimum on the TRAP-part has been found for the first time.

Let $r = \sum_{i=k+1}^{n} x_i$ be the number of ones in the ONEMAX-part of the current solution x. For the solution x where the optimum of the TRAP-part has been found for the first time $n/3 < r < (2/3)n$ holds with probability $1 - e^{-\Omega(n)}$. This implies that this solution is accepted by the algorithm. Later on, only solutions that are optimal with respect to the TRAP-part are accepted and we can follow the ideas in the proof of Theorem 1 to complete the proof. □

The optimization time of the $(1 + 1)$ EA on TRAP-ONEMAX can be reduced significantly using bit-wise neutrality with different numbers of genotypic bits. We use the setting already investigated for the $(1 + 1)$ EA* on NH-ONEMAX and show that this can also help to speed up the computation of the $(1 + 1)$ EA on TRAP-ONEMAX.

Theorem 4 *Using the (1 + 1) EA with $p_{ge} = 1/\ell$ together with the parity encoding where for each x_i, $1 \leq i \leq k$, of the phenotype 2ℓ genotypic bits and for each x_j, $k + 1 \leq j \leq n$, of the phenotype one genotypic bit is used, the expected optimization time on* TRAP-ONEMAX *is $O(n^2 \log n)$.*

Proof Each bit on the TRAP-part is flipped with probability at least $1/e$ and with probability at most $1/2$. Hence, a solution x with k leading ones is found after an expected number of $e^{\ln n} = O(n)$ steps for the first time. To reach an improvement the number of one

in the ONEMAX-part has to be at least $k + 1$. As long the number of ones in the ONEMAX-part is not at least $k + 1$ the probability that the number of ones in this part increases by at least $\ln n + 1$ is at least $e^{-O(\ln n+1)} = \Omega(1/n)$ as the number of flipping bits on the ONEMAX-part is asymptotically Poisson distributed with parameter $\lambda = 1$. Hence, the expected waiting time to produce an optimal solution on the TRAP-part with at least $k + 1$ ones in the ONEMAX-part is $O(n^2)$.

The reach the optimum a search point with k leading ones has to be re-sampled which means that non of the bits in the needle trap flip. The expected waiting time for the event is again $e^{\ln n} = O(n)$. Using the $O(n \log n)$ runtime bound for $(1 + 1)$ EA with mutation rate $1/\ell$ on ONEMAX (Droste et al. 2002) the $O(n^2 \log n)$ bound follows. □

6 Conclusions

We have examined the use of bit-wise neutrality in evolutionary algorithms. In our investigations we have pointed out that there is a direct mapping between genotypic and phenotypic mutation rates and derived simple closed equations for two encodings. Hence, using for each phenotypic bit the same number of genotypic bits only changes the overall mutation rate in the phenotype which can also be achieved by doing this directly without using neutrality. Later on, we have shown that using different numbers of genotypic bits for each phenotypic bit can help to speed up computation. These results are obtained by rigorous runtime analyses on plateau and trap functions that point out that bit-wise neutrality may be useful for hard problems in this case.

A topic for future work is to examine the effect of neutrality for dynamic problems. In such problems neutrality might have different effects that are helpful for the adaptation process. In particular, it may help the algorithm to detect which components of the problem change over time and adapt the mutation rates accordingly.

Acknowledgement We thank Riccardo Poli for an interesting discussion on the topic of this paper.

References

Collins M (2005) Finding needles in haystacks is harder with neutrality. In: Proceedings of the annual conference on genetic and evolutionary computation (GECCO '05). ACM Press, pp 1613–1618

Droste S, Jansen T, Wegener I (1998) A rigorous complexity analysis of the (1 + 1) evolutionary algorithm for separable functions with boolean inputs. Evol Comput 6(2):185–196

Droste S, Jansen T, Wegener I (2002) On the analysis of the (1 + 1) evolutionary algorithm. Theor Comput Sci 276:51–81

Garnier J, Kallel L, Schoenauer M (1999) Rigorous hitting times for binary mutations. Evol Comput 7(2):173–203

Gutjahr WJ, Sebastiani G (2008) Runtime analysis of ant colony optimization with best-so-far reinforcement. Methodol Comput Appl Probab 10(3):409–433

He J, Yao X (2001) Drift analysis and average time complexity of evolutionary algorithms. Artif Intell 127(1):57–85

Huynen MA (1996) Exploring phenotype space through neutral evolution. J Mol Evol 43:165–169

Huynen MA, Stadler P, Fontana W (1996) Smoothness within ruggedness: the role of neutrality in adaptation. Proc Natl Acad Sci USA 93:397–401

Jansen T, Wegener I (2001) Evolutionary algorithms—how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Trans Evol Comput 5(6):589–599

Kimura M (1968) Evolutionary rate at the molecular level. Nature 217:624–626

Neumann F, Sudholt D, Witt C (2007) Comparing variants of MMAS ACO algorithms on pseudo-boolean functions. In: Proceedings of engineering stochastic local search algorithms (SLS '07), LNCS, vol 4638. Springer, pp 61–75

Poli R, López EG (2007) On the effects of bit-wise neutrality on fitness distance correlation, phenotypic mutation rates and problem hardness. In: Proceedings of foundations of genetic algorithms (FOGA '07), pp. 138–164

Rothlauf F (2003) Population sizing for the redundant trivial voting mapping. In: Proceedings of the annual conference on genetic and evolutionary computation (GECCO '03), LNCS, vol 2724. Springer, pp 618–627

Schuster P (2002) Molecular insights into evolution of phenotypes. In: Crutchfield JP, Schuster P (eds) Evolutionary dynamics—exploring the interplay of accident, selection, neutrality and function. Santa Fe Institute Series in the Science of Complexity. Oxford University Press, Oxford

Stephens C, Waelbroeck H (1999) Codon bias and mutability in HIV sequences. J Mol Evol 48:390–397

Toussaint M, Igel C (2003) Neutrality and self-adaptation. Nat Comput 2(2):117–132

Wegener I, Witt C (2005) On the optimization of monotone polynomials by simple randomized search heuristics. Combin Probab Comput 14:225–247

Weicker K, Weicker N (2001) Burden and benefits of redundancy. In: Proceedings of foundations of genetic algorithms (FOGA '00). Morgan Kaufmann, pp 313–333