Redundant Representations in Evolutionary Computation

Franz Rothlauf

Working Paper 3/2003
January 2003

Working Papers in Information Systems

University of Mannheim
Department of Information Systems 1
D-68131 Mannheim/Germany
Phone +49 621 1811691, Fax +49 621 1811692
E-Mail: wifo1@uni-mannheim.de
Internet: `http://www.bwl.uni-mannheim.de/wifo1`

Redundant Representations in Evolutionary Computation

Franz Rothlauf
Dept. of Information Systems 1
University of Mannheim
D-68131 Mannheim/Germany
rothlauf@uni-mannheim.de

David E. Goldberg
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Av. Urbana, IL 61801
deg@illigal.ge.uiuc.edu

January 23, 2003

Abstract

This paper investigates how the use of redundant representations influences the performance of genetic and evolutionary algorithms. Representations are redundant if the number of genotypes exceeds the number of phenotypes. A distinction is made between *synonymously* and *non-synonymously* redundant representations. Representation are synonymously redundant if the genotypes that represent the same phenotype are very similar to each other. Non-synonymously redundant representations do not allow genetic operators to work properly and result in a lower performance of evolutionary search. When using synonymously redundant representations, the performance of selectorecombinative genetic algorithms (GAs) depends on the modification of the initial supply. Theoretical models are developed that show the necessary population size to solve a problem and the number of generations goes with $O(2^{k_r}/r)$, where k_r is the order of redundancy and r is the number of genotypic building blocks (BB) that represent the optimal phenotypic BB. Therefore, uniformly redundant representations do not change the behavior of GAs. Only by increasing r, which means overrepresenting the optimal solution, does GA performance increase. Therefore, non-uniformly redundant representations can only be used advantageously if a-priori information exists regarding the optimal solution. The validity of the proposed theoretical concepts is illustrated for the binary trivial voting mapping and the real-valued link-biased encoding. The empirical investigations show that the developed population sizing and time to convergence models allow an accurate prediction of the empirical results.

1 Introduction

Redundant representations are increasingly being used in evolutionary computation. Redundant representations use a higher number of alleles for encoding phenotypic information in the genotype than is essentially to construct the phenotype. Although the practice of redundancy has steadily increased over the last few years, there is little theory regarding the influence of redundant representations on the performance of genetic and evolutionary algorithms (GEAs).

The purpose of this paper is to examine both theoretically and empirically, how redundant representations affect the performance of genetic and evolutionary algorithms. In particular we classify the different types of redundant representations by distinguishing between *synonymously* and *non-synonymously* redundant representations. Representations are synonymously redundant if the genotypes that represent the same phenotype have similar properties and are next to each other in the mutational space. We describe the problems of evolutionary optimization methods

when using non-synonymously redundant representations, and investigate the effect of synonymously redundant representations on the population size which is necessary to solve a specific problem and on the number of generations until a selectorecombinative GA converges. The theoretical models show that in comparison to non-redundant representations, synonymously redundant representations do not change the performance of selectorecombinative GAs as long as all phenotypes are represented on average by the same number of different genotypes. Only when some phenotypes are overrepresented does the performance of a GA change by the use of a redundant representation. We present quantitative models and illustrate their relevance for two different types of synonymously redundant representations. Firstly, we use the redundant trivial voting mapping for the one-max problem and for the concatenated deceptive trap problem. Secondly, we use the redundant link-biased encoding, which was proposed by Palmer (1994), for the encoding of trees. For this encoding the overrepresentation of some specific solutions can be controlled by an encoding parameter P_1. We can verify that the proposed theoretical concepts accurately predict how the expected GA performance depends on the overrepresentation of some phenotypes.

The paper is structured as follows. In section 2 we take a closer look at the characteristics of redundant representations. We give a short literature overview, discuss the differences between synonymously and non-synonymously redundant representations and explain why non-synonymous redundancy results in problems for evolutionary search. This is followed in section 3 by the population sizing and time-to-convergence model for synonymously redundant representations. Consequently, in section 4 we examine how the use of the synonymously redundant trivial voting mapping for binary problems affects the necessary population size and time to convergence of GAs. Section 5 presents an investigation into the Link-biased encoding. When using this kind of tree representation, the overrepresentation of some specific phenotypes can be controlled by the encoding-specific parameter P_1. This allows us not only to test the validity of the proposed theoretical models for continuous representations, but also to illustrate how GA performance depends on the overrepresentation of some specific phenotypes. In section 6 we present some directions of future work. The paper ends with concluding remarks.

2 Redundant Representations

This section gives an overview of the use of redundant representations and presents a classification of the different types of redundant representations based on their synonymity.

2.1 Redundant Representations and Neutral Networks in Evolutionary Computation

We describe the use of redundant representations in evolutionary computation and review the work that was done in this field over the last few years.

Redundant representations are not an invention of evolutionary computation researchers but are commonly used in nature for the encoding of genetic information. Currently, in biology there are different opinions about the concepts that underly the process of evolution in nature. A common concept is based on selection, whereas other models view evolution as a result of redundant representations and neutral mutations. The theory of selection goes back to Darwin (1859) and assumes that genetic changes are a result of selection combined with variation operators like crossover and random mutations. During the process of evolution the variation operators sometimes result in fitter individuals which gradually replace less-fit individuals in the population.

In contrast, the neutral theory which was proposed by Kimura (1983), assumes that not selection but the random fixation of neutral mutations is the main source of evolution. Kimura observed that

in nature the number of different genotypes which store the genetic material of an individual greatly exceeds the number of different phenotypes which determine the outward appearance. Therefore, the representation which describes how the genotypes are assigned to the phenotypes must be redundant and neutral mutations become possible. A mutation is neutral if its' application to a genotype does not result in a change of the corresponding phenotype. Because large parts of the genotype have no actual effect on the phenotype, evolution can use them as a store for genetic information that was in the past necessary for survival, and as a playground for developing new properties of the individual that can be advantageous in the future. Neutral mutations are the tool for designing these new properties without interfering with the current phenotype of an individual. Although, most of the mutations are neutral, sometimes some have an effect on the phenotype and bring some new genetic material which was developed by neutral mutations into life.

Following the work of Kimura some biological studies (Huynen, Stadler, & Fontana, 1996; Huynen, 1996; Schuster, 1997; Reidys & Stadler, 1998) focused on the neutral theory. These studies showed that the connectivity between fitness landscapes can be increased by the introduction of redundant representations and neutral mutations. Different genotypes which are assigned to the same phenotype (neutral sets) allow a population to move through the search space more easily and to find new advantageous areas of the search space that would not have been accessible without neutral mutations. Surprisingly, the neutral theory became even more popular in the field of genetic and evolutionary computation (Banzhaf, 1994; Dasgupta, 1995). There is great interest in how redundant representations and neutral search spaces influence the behavior, and especially the evolvability of evolutionary algorithms (Barnett, 1997; Barnett, 1998; Shipman, 1999; Shipman, Shackleton, & Harvey, 2000; Shackleton, Shipman, & Ebner, 2000; Shipman, Shackleton, Ebner, & Watson, 2000; Ebner, Langguth, Albert, Shackleton, & Shipman, 2001; Smith, Husbands, & M., 2001c; Smith, Husbands, & M., 2001a; Smith, Husbands, & M., 2001b; Barnett, 2001; Yu & Miller, 2001; Yu & Miller, 2002; Toussaint & Igel, 2002). The general idea behind most of this work is that the evolvability of a population, which is defined as the ability of random variations to sometimes produce improvements, is increased by the use of redundant representations. Furthermore, because redundant representations allow a population to change the genotype without changing the phenotype the ability of a population to adapt after changes and the performance of evolutionary algorithms should increase.

However, in most of this work the focus is not on the performance of GEAs, but on characteristics of the search like reachability of phenotypes, evolvability of populations, or connectivity of search spaces. No results have been presented up till now that clearly indicate the superiority of redundant representations and neutral search on practical test problems or real-world instances. Recently, Knowles and Watson (2002) presented an investigation into the performance of neutral search for NK landscapes, H-IFF, and MAX-SAT problems. The results showed that using arbitrary redundant representations (Random Boolean Network mapping) does not increase the performance of mutation-based search for the considered test problems. In most of the problems used, adding redundancy appeared to reduce performance.

Although, at the moment the focus in investigating the role of redundant representations is mainly on neutral mutations and their effects on search characteristics, there is other work which tries to address the effects of redundancy on the performance of evolutionary search. Researchers used different types of redundant representations and sometimes observed either an increase or a decrease in the performance of evolutionary algorithms (EAs). Over time different opinions regarding the effects of redundancy on the performance of GEAs have been developed. Some work noticed that redundant representations lead to a reduction in EA performance (Davis, 1989; Eshelman & Schaffer, 1991; Ronald, Asenstorfer, & Vincent, 1995). The low performance was argued to be either due to a loss of diversity in the population, or because different genotypes that

represent the same phenotype compete against each other. Also, the larger size of the search space was listed as a reason for lower EA performance.

In contrast, other mostly application-oriented work reports higher performance with additional redundancy (Gerrits & Hogeweg, 1991; Cohoon, Hegde, Martin, & Richards, 1988; Julstrom, 1999), which some researchers ascribe to an increase in diversity that hinders premature convergence. Further work considered the computational implications of genetic code-like representations in gene expression (Kargupta, 2000; Kargupta, 2001). Kargupta investigated how redundant representations influence the energy of the Fourier spectrum. The results show that using redundant representations and encoding phenotypes with higher fitness by a larger number of genotypes results in a higher energy of the Fourier spectrum, reduces the difficulty of the optimization problem, and therefore allows a more effective evolutionary search.

This short literature review has shown that the influence of redundant representations on the performance of evolutionary algorithms is a strongly disputed topic. It can be expected that there is no easy and general answer, and not all types of redundant representations will be useful (Harvey & Thompson, 1997). To find answers, it is necessary to characterize the different types of redundant representations regarding their specific properties, and to develop quantitative models describing how solution quality and run duration of GEAs is influenced. This approach can help to clear up some of the disputed questions and to find out under which circumstances which type of redundant representation can be beneficial for GEAs. Consequently, we develop in the following a classification for different types of redundant representations and develop in section 3 quantitative models.

2.2 Synonymously and Non-Synonymously Redundant Representations

In the following we give some basic definitions and develop a classification for different types of redundant representations. The classification is based on the synonymity of representations.

When talking about representations we have to distinguish between genotypes and phenotypes. Φ_g is defined as the genotypic search space, where the operators crossover and mutation are applied to. Φ_p is the phenotypic search space. The fitness of an individual depends on the properties of the phenotype $x^p \in \Phi_p$. A representation $f_g : \Phi_g \to \Phi_p$ determines which phenotypes $\boldsymbol{x}^p \in \Phi_p$ are represented by which genotypes $\boldsymbol{x}^g \in \Phi_g$. We want to assume that every phenotype x^p is assigned to at least one genotype x^g. Otherwise, if a phenotype x^p is not represented by some genotype x^g this solution can never be found by the used optimization algorithm.

A representation f_g is redundant if the size of the genotypic search space is larger than the size of the phenotypic search space, $|\Phi_g| > |\Phi_p|$. This means, there are more different genotypes than phenotypes. When using search spaces where not all possible phenotypes or genotypes are accessible by the used search method, a representation is redundant if the number of accessible phenotypes is smaller than the number of accessible genotypes. Therefore, in general a representation f_g is redundant if at average one accessible phenotype is represented by more than one genotype. Redundant representations are less efficient encodings which use an additional number of genes but do not increase the amount of relevant information represented. Therefore, a representation is redundant if l different phenotypes are assigned to m different genotypes where $m > l$. Although the larger number of possible genotypes would allow us to encode more individuals than there are phenotypes, some of the information that exists in the genotypes is not considered.

To classify different types of redundant representations we want to measure how similar the genotypes are that are assigned to the same phenotype. A representation is defined to be synonymously redundant if the genotypes that are assigned to the same phenotype are similar to each other. Consequently, we denote a representation to be non-synonymously redundant if the genotypes that are assigned to the same phenotype are not similar to each other. Therfore, the

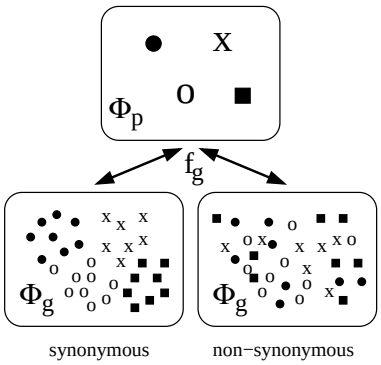


Figure 1: Synonymous versus non-synonymous redundancy. The different symbols indicate different genotypes and their corresponding phenotypes. When using synonymously redundant representations (left), genotypes that represent the same phenotype are similar to each other. When using non-synonymously redundant representations (right), genotypes that represent the same phenotype are not similar to each other but distributed over the whole search space.*put there an f_g on the arrows*

synonymity of a represenation depends on the metric that is defined on Φ_g and Φ_p. The metric defined on Φ_p depends on the properties of the considered problem and the metric defined on Φ_g depends on the used search operator. Depending on different operators and metrics used on Φ_g we get different synonymity of the representation f_g. In Figure 1 we illustrate the differences between synonymously and non-synonymously redundancy. For this illustrative example we use the Euclidean distance between the individuals for indicating how similar different individuals are.

We want to formalize the classification into synonymously and non-synonymously redundant representations. In general, a redundant representation f_g assigns a phenotype x^p to a set of different genotypes $x^g \in \Phi_g^{x_p}$, where $\forall x^g \in \Phi_g^{x_p} : f_g(x^g) = x_p$. All genotypes x^g in the genotypic set $\Phi_g^{x_p}$ represent the same phenotype x^p. A representation is synonymously redundant if the genotypic distances between all $x^g \in \Phi_g^{x_p}$ are small for all different x_p. Therefore, if for all phenotypes the sum over the distances between all genotypes that correspond to the same phenotype

$$\left(\sum_{x^p} \frac{1}{2} \left(\sum_{x^g \in \Phi_g^{x_p}} \sum_{y^g \in \Phi_g^{x_p}} d(x^g, y^g) \right) \right), \tag{1}$$

where $x^g \neq y^g$, is reasonably small a representation is denoted to be synonymously redundant. $d(x^g, y^g)$ depends on the mutation operator used and measures the distance between two genotypes $x^g \in \Phi_g^{x_p}$ and $y^g \in \Phi_g^{x_p}$ which both represent the same phenotype x^p. The distance between two genotypes depends on their genotypic similarity and is small if the two genotypes are similar.

The synonymity of redundant redundant representations is related to the locality of non-redundant representations. A genotype x^g is a neighbor to some other genotype y^g if the distance $d(x^g, y^g) = d_{min}$, where $d_{min} \neq 0$ is the minimal distance between two individuals in the genotypic search space. When using binary representations, $d_{min} = 1$ and two genotypes are neighbors if they differ in one allele. The locality of a representation describes if neighboring genotypes correspond to neighboring phenotypes. If neighboring genotypes correspond to neighboring phenotypes, a representation has high locality and small changes in a genotype result in small changes in the corresponding phenotype. There is evidence, both analytical (for example Whitley (1999) for mutation-based search or Rothlauf (2002) for crossover-based search), and empirical (for example

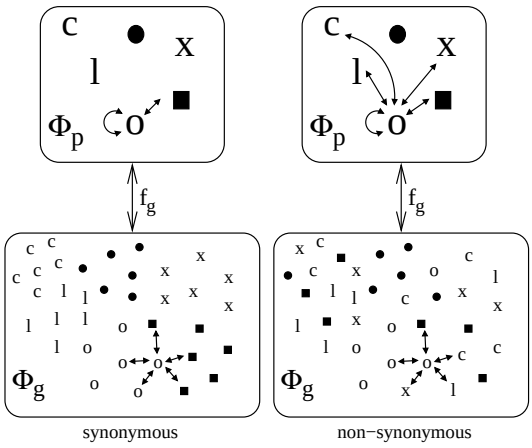


Figure 2: The effects of small mutation steps for synonymously versus non-synonymously redundant representations. The different symbols indicate different genotypes and their corresponding phenotypes. The arrows indicate neighboring individuals. When using synonymously redundant representations a mutation results in either the same or a similar phenotype. Contrastly, when using non-synonymously redundant representations the mutation of a genotype results in completely different phenotypes.

Gottlieb et al. (2001) or Rothlauf and Goldberg (2000)), that shows for easy problems low locality representations result in low GEA performance. The genetic operators mutation and crossover no longer work properly as they create new offspring that are not similar to their parent(s). When using low-locality representations, guided GEA-search becomes random search. Therefore, low locality representations are a necessity for efficient evolutionary search.

These concepts can also be applied to redundant representations. When using non-synonymously redundant representations genetic operators like mutation or crossover can result in an offspring that is phenotypically completely different from its' parent(s). Using neutral search spaces where the connectivity between the phenotypes is strongly increased by the use of a redundant representation allows us to reach many different phenotypes by one single mutation step. However, increasing the connectivity between the phenotypes results in random search and decreases the efficiency of EAs. It has the same effect as when using large mutation steps for non-redundant representations. By using large mutation steps many different individuals can be reached in one step and the connectivity of the phenotypes increases. As a result, guided search becomes more difficult and search becomes more random. Therefore, we expect reduced EA performance on easy problems when using non-synonymously redundant representations. Examples for non-synonymously redundant representations are the direct binary mapping, the cellular automaton mapping or the random boolean network mapping which have been proposed by Shackleton et al. (2000). The use of these types of representations strongly increases the connectivity between phenotypes what results in a more randomized search. Consequently, Knowles and Watson (2002) have shown for some test problems that the performance of mutation-based search using the Random Boolean Network mapping decreases.

In contrast, when using synonymously redundant representations, the connectivity between the phenotypes is not increased. Small genotypic variations can not result in large phenotypic changes but result either in the same or a similar phenotype. Figure 2 illustrates this behavior and compares it to non-synonymously redundant representations. Examples for synonymously redundant representations are the trivial voting mapping (Shackleton et al., 2000) which is investigated more

closely in section 4.

3 Population Sizing and Time to Convergence for Synonymously Redundant Representations

In the previous subsection we have suggested how non-synonymously redundant representations result in randomized search as they increase the connectivity of the search space. Therefore, we want to focus in the following on synonymously redundant representations and develop models that describe their influence on the performance of selectorecombinative GAs.

3.1 Characteristics of Redundant Representations

In this subsection we introduce some quantities that can be used for characterizing the properties of synonymously redundant representations. We use the definitions from subsection 2.2.

To describe a redundant representation, we introduce the order of redundancy k_r. k_r is defined as $\log(|\Phi_g|)/\log(|\Phi_p|)$ and measures the amount of redundant information in the encoding. When using binary genotypes and binary phenotypes, the order of redundancy can be calculated as

$$k_r = \frac{\log(2^{l_g})}{\log(2^{l_p})},$$

where l_g is the length of the binary genotype and l_p is the length of the binary phenotype. When using a non-redundant representation, the number of genotypes equals the number of phenotypes and $k_r = 1$.

Furthermore, we want to characterize not only how much a representation is redundant, but also how a representation is redundant. We are especially interested in the overrepresentation and underrepresentation of specific solutions. Therefore, we introduce r as the number of genotypes that represent the one phenotype that has the highest fitness. When using non-redundant representations, every phenotype is assigned to exactly one genotype and $r = 1$. However, in general, $1 \leq r \leq |\Phi_g| - |\Phi_p| + 1$.

In the following we want to focus on how redundant representations influence the behavior of selectorecombinative GAs. Selectorecombinative GAs use crossover as the main search operator and mutation only serves as a background operator. When focusing on selectorecombinative GAs we implicitly assume that there are building blocks (BBs) and that the GA process schemata. Consequently, we must define how k_r and r depends on the properties of the BBs.

In general, when looking at BBs of size k there are 2^k different phenotypic BBs which are represented by 2^{kk_r} different genotypic BBs. Therefore,

$$k_r = \frac{k_p}{k_g},$$

where k_g denotes the genotypic size of a BB and k_p the size of the corresponding phenoypic BB. As before, a representation is redundant if $k_r > 1$. The size of the genotypic BBs is k_r times larger than the size of the phenotypic BB. Furthermore, r is defined as the number of genotypic BBs of length kk_r that represent the best phenotypic BB of size k. Therefore, in general,

$$r \in \{1, 2, \ldots, 2^{kk_r} - 2^k + 1\}.$$

In contrast to k_r which is determined by the representation used, r depends not only on the used representation, but also on the specific problem that should be solved. Different instances of a

x^g	x^p
00 00, 00 01, 01 00, 01 01	0 0
10 00, 10 01, 11 00, 11 01	1 0
00 10, 01 11, 00 11, 01 11	0 1
10 10, 10 11, 11 10, 11 11	1 1

Table 1: An example of a uniformly redundant representation, where $k_r = 2$ and $r = 4$

problem result in different values of r. If we assume that k_r is an integer (each phenotypic allele is represented by k_r genotypic alleles) the possible values of the number of genotypic BBs that represent the optimal phenotypic BB can be calculated as

$$r = i^k, \text{ with } i \in [1, 2, \ldots, 2^{k_r} - 1]. \tag{2}$$

A representation is uniformly redundant if all phenotypes are represented by the same number of different genotypes. Therefore, when using an uniformly redundant representation every phenotypic BB of size $k = k_p$ is represented by

$$r = 2^{k(k_r - 1)} \tag{3}$$

different genotypic BBs.

Table 1 gives an example for a uniformly redundant encoding. Two bits in a phenotype x^p are represented by four bits in the genotype x^g. Therefore, $k_r = 2$ and $r = 4$. With $|\Phi_p| = 2^k = 2^2$ the size of the genotypic space is $|\Phi_g| = 2^{kk_r} = 2^4 = 16$.

By introducing redundancy the search space for a GA using binary phenotypes of string length $l = l_p$ is increased from $|\Phi_p| = 2^l$ to $|\Phi_g| = 2^{lk_r}$. The length of the individuals increases from $l = l_p$ in the phenotypic space to $l_g = k_r * l$ in the genotypic space. To represent all phenotypes, each individual $\boldsymbol{x}_p \in \Phi_p$ must be represented by at least one genotype $\boldsymbol{x}_g \in \Phi_g$. If $|\Phi_g| = |\Phi_p|$, and each phenotype is represented by at least one genotype, we have a non-redundant, one-to-one mapping.

3.2 Population Sizing

As we focus in our investigation on selectorecombinative GAs we can use the existing theory describing the behavior of selectorecombinative GAs from Harik, Cantú-Paz, Goldberg, and Miller (1997), and Thierens and Goldberg (1993). They describe for non-redundant representations how the population size and the time to convergence that is necessary to solve a specific problem depend on the characteristics of the problem. In the following, we use these models for describing the effects of synonymously redundant representations on the performance of GAs.

Following Harik, Cantú-Paz, Goldberg, and Miller (1997) the probability that a GA with a population size N converges after t_{conv} generations to the correct solution is

$$P_n = \frac{1 - (q/p)^{x_0}}{1 - (q/p)^N},$$

where x_0 is the expected number of copies of the best BB in the randomly initialized population, $q = 1 - p$, and p is the probability of making the right choice between a single sample of each BB

$$p = \mathbb{N}\left(\frac{d}{\sqrt{2m'}\sigma_{BB}}\right). \tag{4}$$

d is the signal difference between the best BB and its strongest competitor, $m' = m-1$ with m is the number of BBs in the problem, σ^2_{BB} is the variance of a BB, and $q = 1-p$ is the probability of making the wrong decision between two competing BBs. It has been shown in Harik, Cantú-Paz, Goldberg, and Miller (1997) that this random walk or Gambler's ruin model can be used for describing the behavior of selectorecombinative GAs propagating schemata and BBs. In the following, this model is the basis for describing the influence of synonymously redundant representations on the behavior of GAs.

For a randomly initialized population with no redundancy, $x_0 = N/2^k$. The situation changes when using redundant representations. Then, the initial supply depends on the characteristics of the representation, namely r and k_r. With r the number of genotypic BBs of length kk_r that represent the best phenotypic BB of length k, we get

$$x_0 = N\frac{r}{2^{kk_r}}, \tag{5}$$

where k_r is the order of redundancy. The assumption that redundant representations affect the initial supply of BBs is the core idea behind the proposed model describing the influence of synonymously redundant representations on GA performance. We assume that other effects of synonymously redundant representations on GA performance can be neglected. Consequently, when using uniformly redundant representations, $r = 2^{k(k_r-1)}$ and $x_0 = N/2^k$. These are the same value as when using non-redundant representations. Therefore, GA performance does not change when using uniformly redundant representations.

As the variance σ^2_{BB} and the number of BBs, m is not affected by the use of a redundant representation, the probability of GA failure $\alpha = 1 - P_n$ can be calculated as

$$\alpha = 1 - \frac{1-(q/p)^{x_0}}{1-(q/p)^N}. \tag{6}$$

If we assume that x_0 is small and $q < p$ we can assume that $1-(q/p)^N$ converges faster to 1 than $1-(q/p)^{x_0}$. Using these approximations (see also Harik et al. (1997)) the equation can be simplified to

$$\alpha \approx \left(\frac{1-p}{p}\right)^{x_0}.$$

Therefore, we get for the population size

$$N \approx \frac{2^{kk_r}}{r}\left(\frac{\ln(\alpha)}{\ln\left(\frac{1-p}{p}\right)}\right). \tag{7}$$

The normal distribution in equation 4 can be approximated using the first two terms of the power series expansion (see Abramowitz and Stegun (1972)) as $\mathbb{N}(x) \approx 1/2 + x/2$. Substituting p from equation 4 into equation 7 we get:

$$N \approx \frac{2^{kk_r}}{r}\ln(\alpha)/\ln\left(\frac{1-x}{1+x}\right),$$

where $x = d/\sqrt{2m'}\sigma_{BB}$. Since x is a small number, $\ln(1-x)$ can be approximated with $-x$ and $\ln(1+x)$ with x. Using these approximations we finally get for the population size N:

$$N \approx -\frac{2^{k_r k-1}}{r}\ln(\alpha)\frac{\sigma_{BB}\sqrt{\pi m'}}{d}. \tag{8}$$

The population size N goes with $O\left(\frac{2^{k_r}}{r}\right)$ when using synonymously redundant representations. With increasing r the number of individuals that are necessary to solve a problem decreases. Using a uniformly redundant representation, where $r = 2^{k(k_r-1)}$, does not change the population size N in comparison to non-redundant representations.

3.3 Run Duration

To describe the performance of GAs, we must calculate not only the number of individuals that are necessary for solving a problem, but also the expected number of generations until convergence.

Based on Mühlenbein and Schlierkamp-Voosen (1993) and Thierens and Goldberg (1993), Miller and Goldberg developed a convergence model for selectorecombinative GAs (Miller & Goldberg, 1996b; Miller & Goldberg, 1996a). The convergence time t_{conv} depends on the length of the phenotypes $l = l_p$ and the used selection scheme. Using the selection intensity I the convergence model is

$$p(t) = 0.5\left(1 + \sin\left(\frac{It}{\sqrt{l}} + \arcsin(2p(0) - 1)\right)\right),$$

where $p(0) = x_0/N$ is the proportion of best building blocks in the initial population. I depends only on the used selection scheme. The number of generations t_{conv} it takes to fully converge the population can be calculated by putting $p(t_{conv}) = 1$:

$$t_{conv} = \frac{\sqrt{l}}{I}\left(\frac{\pi}{2} - \arcsin\left(2p(0) - 1\right)\right). \tag{9}$$

If we assume $k = 1$ and uniform redundancy (equal proportion of 1s and 0s in the initial population) we get $p(0) = 0.5$. Then, the number of generations until convergence simplifies to

$$t_{conv} = \frac{\pi}{2}\frac{\sqrt{l}}{I}.$$

With redundancy the initial proportion of building blocks is $p(0) = \frac{r}{2^{kk_r}}$ (see equation 5). Using $\arcsin(x) = x + o(x^3)$ the time until convergence could be approximated by

$$t_{conv} = \frac{\sqrt{l}}{I}\left(1 + \frac{\pi}{2} - \frac{r}{2^{k_r k-1}}\right). \tag{10}$$

With increasing $r/2^{k_r}$ the time to convergence t_{conv} is reduced. Therefore, the optimal solution is found after a lower number ob generations if it is overrepresented by the synonymously redundant representation. For uniform redundancy $r = 2^{k(k_r-1)}$, we get

$$t_{conv} = \frac{\sqrt{l}}{I}\left(1 + \frac{\pi}{2} - \frac{1}{2^{k-1}}\right).$$

The time until convergence when using uniformly redundant representations is the same as without redundancy.

3.4 Overall Problem Complexity

After we have calculated the number of individuals that are necessary for solving a problem (see equation 8), and the number of generations that GAs using only crossover need to converge (see

equation 10), we can calculate the absolute number of fitness calls that are necessary for solving a problem:

$$N * t_{conv} \approx -\frac{2^{k_r k-1}}{r} \ln(\alpha) \frac{\sigma_{BB}\sqrt{\pi m'}}{d} * \frac{\sqrt{l}}{I} \left(1 + \frac{\pi}{2} - \frac{r}{2^{k_r k-1}}\right) =$$
$$= \frac{\sqrt{\pi l m'}}{I} \ln(\alpha) \frac{\sigma_{BB}}{d} \left(1 - \frac{2^{k k_r}}{4r}(2+\pi)\right)$$

The overall number of fitness calls goes with $O(2^{k_r}/r)$. In comparison to non-redundant representations the number of fitness calls stays constant for synonymously redundant representations if $r = 2^{k(k_r-1)}$. Then $x_0/N = 1/2^k$ and the representation is uniformly redundant.

4 Trivial Voting Mapping for Binary Encoded Problems

In the previous section we developed theoretical models describing how synonymously redundant representations influence the solution quality and the time that is necessary to find the good solutions. In the following section we investigate if the proposed models allow a good prediction of GA performance for the trivial voting (TV) mapping. The TV mapping is a synonymously redundant representation and we use it for one-max and concatenated deceptive trap problems. During our investigation we are particularly interested in whether the developed models allow us to accurately predict the expected solution quality and running time of a selectorecombinative GA.

4.1 The Trivial Voting Mapping

In the following subsection we give a short introduction into the trivial voting mapping.

When using the TV mapping, a set of mostly consecutive, genotypic alleles is relevant for the value of one allele in the phenotype. Each allele in the genotype can only influence the value of one allele in the phenotype. The value of the phenotypic allele is determined by the majority of the values in the genotypic alleles. In general, the different sets of alleles in the genotype defining one phenotypic allele have the same size. The TV mapping is a synonymously redundant representation as all genotypes that represent the same phenotype are very similar to each other. A mutation in a genotype results either in the same corresponding phenotype, or in one of its neighbors.

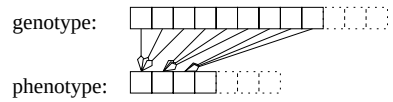


Figure 3: The trivial voting mapping

The TV mapping can easily be characterized using the representation parameters defined in subsection 3.1. The order of redundancy k_r is simply the number of genotypic alleles that determine the value of one phenotypic allele. Figure 3 gives an example for the TV mapping.

Shackleton, Shipman, and Ebner (2000) applied the TV mapping to binary strings in the context of the neutral theory. When used for binary strings, binary genotypes $x^g \in \mathbb{B}^{l_g}$ are assigned to binary phenotypes $x^p \in \mathbb{B}^{l_p}$. The length of a genotype is larger than the length of a phenotype, $l_g > l_p$. The value of one phenotypic bit is determined by the majority of the values in the corresponding genotypic bits (majority vote). However, if k_r is even then the number of ones could equal the number of zeros. Therefore, half the cases that result in a tie should encode a one in the corresponding phenotypic allele, and half the cases should represent a zero. For example, for $k_r = 4$ the genotypic BBs 1100, 1010, and 1001 represent a 1 and the phenotypic BBs 0011, 0101, 0110 represent a zero.

Because the majority of the votes determines the values of the corresponding phenotypic allele, the TV mapping is a uniformly redundant representation. Each phenotypic BB is represented by the same number of genotypic BBs which is $2^{k(k_r-1)}$, where k is the size of the phenotypic BB.

As we are not only interested in uniformly redundant representations, but also want to know how non-uniformly redundant representations influence GA performance, we extend the TV mapping to allow the encoding to overrepresent some individuals. Therefore, we want to assume that if the number of ones in the genotypic alleles $x^g_{k_r i+j}$, where $i \in \{0,\ldots,l_p-1\}$ and $j \in \{0,\ldots,k_r-1\}$, is larger or equal than a constant u then the value of the phenotypic allele x^p_i is set to one. Vice versa, the phenotypic allele x^p_i is set to zero if less than u of the corresponding genotypic alleles are set to one. Therefore,

$$x^p_i = \begin{cases} 0 \text{ if } & \sum_{j=0}^{k_r-1} x^g_{k_r i+j} < u \\ 1 \text{ if } & \sum_{j=0}^{k_r-1} x^g_{k_r i+j} \geq u, \end{cases}$$

where $u \in \{1,\ldots,k_r\}$. x^g_i respectively x^p_i denotes the ith allele of the genotype respectively phenotype. u can be interpreted as the number of genotypic alleles that must be set to one to encode a one in the corresponding phenotypic allele. We denote this representation the extended trivial voting (eTV) mapping. For $u = (k_r+1)/2$ (k_r must be odd) we get the original TV mapping. Extending the TV mapping in the proposed way allows us to investigate how non-uniform redundancy influences the performance of GAs.

When using the eTV mapping, the number r of genotypic BBs that can represent the optimal phenotypic BB depends on the number of ones in the genotypic alleles that determine the value of the corresponding phenotypic allele. Considering equation 2 we get

$$r = \left(\sum_{j=u}^{k_r} \binom{k_r}{j} \right)^k, \tag{11}$$

where $u \in \{1,\ldots,k_r\}$. k denotes the size of the phenotypic BB. We want to give a short illustration. We use a redundant representation with $k_r = 3$, $k = 1$, and the optimal BB is $x^p_i = 1$ (compare Figure 3). Because $u \in \{1,\ldots,k_r\}$ there are three different values possible for r. For $u = 1$ the phenotypic allele x^p_i is set to one if at least one of the three corresponding genotypic alleles $x^g_{ik_r}$, $x^g_{ik_r+1}$, or $x^g_{ik_r+2}$ is set to one. Therefore, a one in the phenotype is represented by $r = \sum_{j=1}^{3} \binom{k_r}{j} = 7$ different genotypic BBs (111, 110, 101, 011, 100, 010, and 001). For $u = 2$, the optimal genotypic BB $x^p_i = 1$ is represented by $r = \sum_{j=2}^{3} \binom{k_r}{j} = 4$ different genotypic BBs (111, 110, 101, and 011) and the representation is uniformly redundant. For $u = 2$ we get the original TV mapping. For $u = 3$, the optimal phenotypic BB is represented only by one genotypic BB (111).

4.2 Experiments and Empirical Results

We present empirical results when using the binary trivial voting mapping for the one-max problem and the concatenated deceptive trap problem.

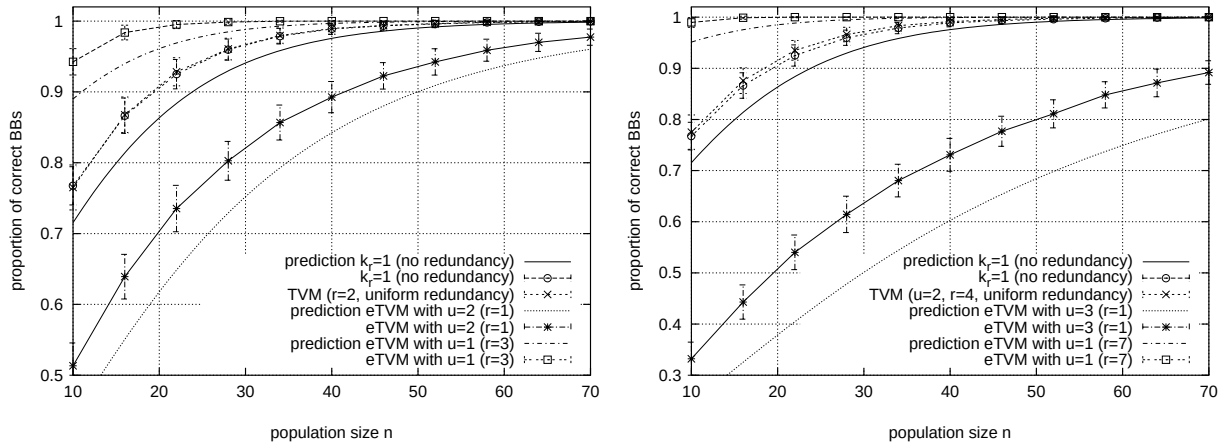
4.2.1 One-Max Problem

The first test example for our empirical investigation is the one-max problem. This problem is very easy to solve for GEAs as the fitness of an individual is simply the number of ones in the binary phenotype. To ensure that recombination results in a proper mixing of the BBs, we use uniform crossover for all experiments with the one-max problem. Furthermore, in all runs we use tournament selection without replacement and a tournament size of 2. For the one-max function the signal difference d equals 1, the size k of the building blocks is 1, and the variance of a building block $\sigma^2_{BB} = 0.25$.

x_i^p	$x_{2i}^g x_{2i+1}^g$ (with $k_r = 2$)		
	extended TV		original TV
	$r = 1$	$r = 3$	$r = 2$
0	00, 01, 10	00	00, 01
1	11	01, 10, 11	10, 11

Table 2: The trivial voting mapping for $k_r = 2$

When using the binary TV mapping for the one-max problem each bit of a phenotype $x^p \in \Phi_p$ is represented by k_r bits of the genotype $x^g \in \Phi_g$. The string length of a genotype x^g is $l_g = k_r * l_p$ and the size of the genotypic search space is $|\Phi_g| = 2^{k_r l_p}$. Table 2 illustrates for $k_r = 2$ the two possibilities ($r = 1$ and $r = 3$) of assigning genotypic BBs $\{00, 01, 10, 11\}$ to one of the phenotypic BBs $\{0, 1\}$ when using the extended TV mapping described in the previous paragraphs. With denoting x_i^p the value of the ith bit in the phenotype, the $2i$th and $(2i + 1)$th bit of a genotype determine x_i^p. Because the size of the BBs $k = 1$, the number of genotypic BBs that represent the optimal phenotypic BB is either $r = 1$ or $r = 3$ (compare equation 11). Furthermore, Table 2 also lists the case where $r = 2$. This case is the original uniformly redundant TV mapping. The second bit of each genotypic BB does not contribute to the construction of the phenotype.



(a) $k_r = 2$ (b) $k_r = 3$

Figure 4: Experimental and theoretical results of the proportion of correct BBs on a 150-bit one-max problem using the trivial voting mapping for $k_r = 2$ (left) and $k_r = 3$ (right). The lines without line points show the theoretical predictions. When using non-uniformly redundant representations, GA performance is changed with respect to the overrepresentation or underrepresentation respectively of the high-quality BBs.

In Figure 4(a) ($k_r = 2$) and Figure 4(b) ($k_r = 3$), the proportion of correct BBs at the end of a run for a 150 bit one-max problem using the TV mapping is shown. For this problem 2^{150} different phenotypes are represented by either 2^{300} ($k_r = 2$) or 2^{450} ($k_r = 3$) different genotypes. If we use the eTV mapping (indicated in the plots as eTVM) we can set u either to 1 or 2 ($k_r = 2$) or to 1, 2, or 3 ($k_r = 3$). The corresponding values for r which can be calculated according to equation 11 as

		extended TV mapping			original TV mapping
		$u = 1$	$u = 2$	$u = 3$	
$k_r = 2$	r	3	1	-	2
	x_0/N	3/4	1/4	-	$2/4 = 1/2$
$k_r = 3$	r	7	4	1	4
	x_0/N	7/8	$4/8 = 1/2$	1/8	$2/4 = 1/2$

Table 3: Properties of the different TV mappings for the one-max problem ($k = 1$)

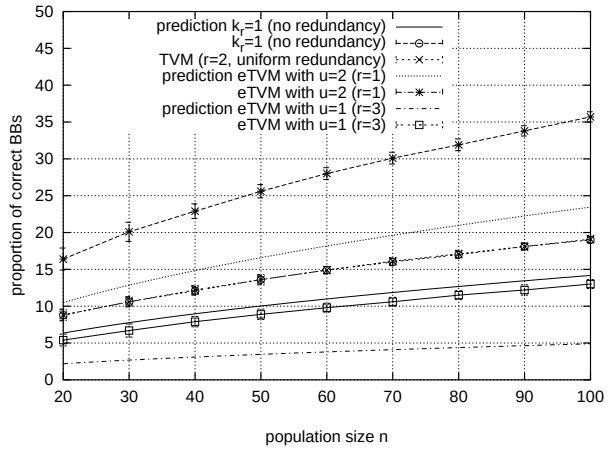
well as x_0/N are shown in Table 3. x_0 is the expected number of copies of the best BB in the initial population and N is the population size. Furthermore, the figures show the results when using the original, uniformly redundant TV mapping, and when using the non-redundant representation with $k_r = 1$. The lines without line points show the theoretical predictions from equation 6, and the lines with line points show the empirical results which are averaged over 250 runs. The error bars indicate the standard deviation.

The results show that for the uniformly redundant TV mapping, $r = 2$ ($k_r = 2$) or $r = 4$ ($k_r = 3$), we get the same performance as for using the non-redundant representation ($k_r = 1$). As in the original model proposed by Harik, Cantú-Paz, Goldberg, and Miller (1997) the theoretical model slightly underestimates GA performance. As predicted by our model which we proposed in subsection 3.2, GA performance does not change when using a uniformly redundant representation. Furthermore, we can see that if the optimal BB is underrepresented ($u = 2$ for $k_r = 2$ and $u = 3$ for $k_r = 3$) GA performance decreases. Equation 6 gives us a good prediction for the expected solution quality if we consider that the non-uniform redundancy of the representation changes the initial BB supply according to equation 5. If the optimal solution is overrepresented ($u = 1$ for both cases, $k_r = 2$ and $k_r = 3$) GA performance increases. Again the theoretical models give a good prediction for the expected proportion of correct BBs.

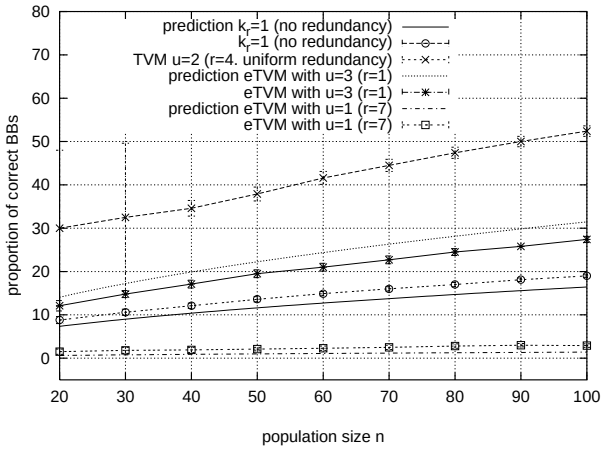
Summarizing the results, we can see that using the uniformly redundant TV mapping does not change GA performance in comparison to using the non-redundant representation. Only if we overrepresent the optimal phenotypic BB, does GA performance increase; likewise, if we underrepresent the optimal BB, GA performance drops. As our derived model is able to make accurate predictions for the expected solution quality, our assumption that synonymously redundant representations influence GA performance by changing the initial supply seems to be valid.

In the remaining paragraphs we perform an empirical investigation into the effect of the TV mapping on the number of generations until the population of a selectorecombinative GA converges. Again we use the one-max problem and the TV mapping from above with the same parameters except the population size is set to $N = 2l_p$ to allow reliable decision making for the one-max problem (Goldberg, Deb, & Clark, 1992). As we use tournament selection without replacement of size two the selection intensity $I = 1/\sqrt{\pi}$.

Figure 5(a) ($k_r = 2$) and Figure 5(b) ($k_r = 3$) show the number of generations that are necessary until 90% of all phenotypic BBs are found over the problem size which is equal to $l = l_p$. The lines without line points show the predictions from equation 9 and the lines with line points plot the empirical results. We can see that the run duration of a GA when using the non-redundant representation ($k_r = 1$) is exactly the same as when using the uniformly redundant TV mapping with $k_r = 2$. For $k_r = 3$ and $u = 2$ (uniform redundancy) the run duration is slightly increased in comparison to the non-redundant encoding. We expect that this difference increases with larger k_r. In agreement with the results from Thierens and Goldberg (1993), we report a



(a) $k_r = 2$



(b) $k_r = 3$

Figure 5: Theoretical predictions and experimental results for the number of generations that are necessary until 90% of all phenotypic BBs are correctly identified. The plots are for one-max problems of different size l and trivial voting mapping with $k_r = 2$ (left) and $k_r = 3$ (right).

small underestimation of the expected number of generations when using either non-redundant, or uniformly redundant, representations.

When using non-uniformly redundant variants of the eTV mapping the underestimation is larger, but nevertheless the model gives a good approximation for the expected number of generations. We can see that with increasing r the run duration increases. For example, if each phenotypic bit is represented by three genotypic bits ($k_r = 3$) and a one is represented if at least one out of three genotypic bits is set to one ($u = 1$) then a GA finds the good solutions after very short time (compare eTVM with $u = 1$). The expected number of generations shows the predicted behavior. The necessary number of generations increases by about $O(\sqrt{l})$. We see that the proposed model allows us to make good predictions for the expected run duration.

4.2.2 Concatenated Deceptive Trap Problem

Our second test example uses deceptive trap functions.

Traps were first used by Ackley (1987) and investigations into the deceptive character of these functions were provided by (Deb & Goldberg, 1993). Figure 6 depicts a 3-bit deceptive trap problem where the size of a BB is $k = 3$. The fitness value of a phenotype x^p depends on the number of ones u in the string of length l. The best BB is a string of l ones which has fitness l. Standard EAs are misled to the deceptive attractor which has fitness $l - 1$. For the 3-bit deceptive trap the signal difference d is 1, and the fitness variance equals $\sigma^2_{BB} = 0.75$. We construct a test problem for our investigation by concatenating $m = 10$ of the 3-bit traps so we get a 30-bit problem. The fitness of an individual x is calculated as $f(x) = \sum_{i=0}^{m-1} f_i(u)$, where $f_i(u)$ is the fitness of the ith 3-bit trap function from Figure 6. Although this function is difficult for GEAs it can be solved with proper population size N.

For deceptive traps of size $k = 3$ we can calculate the number r of genotypic BBs that represent the optimal genotypic BBs according to equation 11. Table 4 summarizes for the modified TV

		extended TV mapping			original TV mapping
		$u = 1$	$u = 2$	$u = 3$	
$k_r = 2$	r	$3^3 = 27$	$1^3 = 1$	-	$2^3 = 8$
	x_0/N	27/64	1/64	-	$8/64 = 1/8$
$k_r = 3$	r	$7^3 = 343$	$4^3 = 64$	$1^3 = 1$	$4^3 = 64$
	x_0/N	343/512	$64/512 = 1/8$	1/512	$64/512 = 1/8$

Table 4: Properties of the different TV mappings for the deceptive trap of size $k = 3$

mapping how r and x_0/N depends on u, which describes how many of the genotypic alleles must be set to 1 to encode a 1 in the phenotype. x_0 is the expected number of copies of the best BB in the initial population and N is the population size. Furthermore, we list the properties of the original uniformly redundant TV mapping.

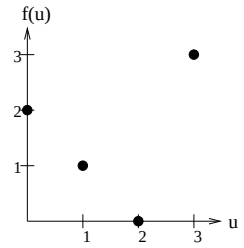


Figure 6: A 3-bit deceptive trap problem

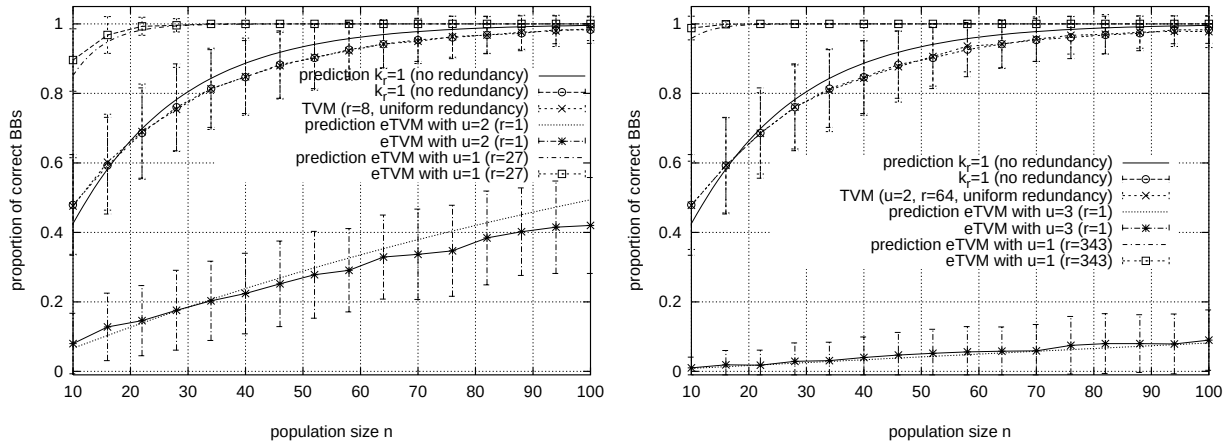
By analogy to the previous paragraphs in Figure 7(a) ($k_r = 2$) and Figure 7(b) ($k_r = 3$) we show the proportion of correct BBs at the end of a run over different population sizes for ten concatenated 3-bit deceptive trap problems. In this problem, 2^{30} different phenotypes are represented by either 2^{60} ($k_r = 2$) or 2^{90} ($k_r = 3$) different genotypes. As before, we use tournament selection without replacement of size 2. In contrast to the one-max problem, two-point crossover was chosen for recombination. Uniform crossover would result in an improper mixing of the BBs because the genotypic BBs are either of length $l_g = k_r l_p = 6$ ($k_r = 2$), or of length $l_g = 9$ ($k_r = 3$). Again, the lines without line points show the predictions of the proposed model for different r. Furthermore, empirical results which are averaged over 250 runs, are shown for various values of r. The results show that for the uniformly redundant TV mapping we get the same performance as when using the non-redundant representation ($k_r = 1$). As in the experiments for the one-max problem the proposed model predicts the experimental results well if the eTV mapping is used and some BBs are underrepresented or overrepresented.

The presented results show that the effects of synonymously redundant representations like the TV mapping on the performance of GEAs can be explained well by a change of the initial supply of high-quality BBs. If the eTV mapping favors high-quality BBs then the performance of GAs is increased. If good BBs are underrepresented the performance is reduced. If the representation is uniformly redundant, GAs show the same performance as when using the non-redundant encoding.

5 Link-biased Encoding for Encoding Trees

In this section we want to illustrate how the theoretical insights from section 3 can be used for predicting the influence of the redundant link-biased encoding on the performance of GEAs. The link-biased encoding is a real-valued representation from the class of weighted encodings and was developed by Palmer (1994).

The purpose of this section is twofold. Firstly, we want to apply the results from section 3 to real-valued representations and show that the approach proposed therein is not only valid for discrete, binary representations (as illustrated in the previous section), but also holds true for continuous representations. Continuous representations using real-valued alleles are redundant by definition if they are used for the representation of permutation problems or other problems with a



(a) $k_r = 2$ (b) $k_r = 3$

Figure 7: Experimental and theoretical results of the proportion of correct BBs for ten concatenated 3-bit deceptive traps. We show results for different variants of the TV mapping and $k_r = 2$ (left) and $k_r = 3$ (right). The lines without line points show the theoretical predictions. As predicted, GA performance sharply decreases if the eTV mapping underrepresents the optimal BB.

finite number of phenotypes. Therefore, it is important to have models available that explain the effects of real-valued representations on GA performance if these representations are non-uniformly redundant. Secondly, we want to show that by using and combining existing population sizing models we are able to quite accurately predict the expected solution quality. Therefore, predicting expected GA performance for the link-biased encoding is an example of how the systematic and theory-guided use of existing models allows accurate modeling of GA behavior.

The more general variant of the link-biased encoding, the link-and-node-biased (LNB) encoding, is redundant and overrepresents some phenotypes. The structures of the solutions that are overrepresented depend on representation-specific parameters. The encoding was proposed to overcome problems of other tree representations like characteristic vectors, predecessor representations, or Prüfer numbers. Abuali, Wainwright, and Schoenefeld (1995) compared the LNB encoding to some other representations for the probabilistic minimum spanning tree (PMST) problem and in some cases found the best solutions by using the LNB encoding. Later, Raidl and Julstrom (2000) proposed a variant of this encoding and observed solutions superior to those of several other representations for the degree-constrained minimum spanning tree problem. For the same type of problem Krishnamoorthy and Ernst (2001) proposed another version of the LNB encoding.

In the following subsection we give a short overview of the functionality of the link-biased encoding. In subsection 5.2 we focus on the redundancy of the representation and illustrate how it depends on the representation-specific parameter P_1. In subsection 5.3 we review a population sizing model which can be used for the link-biased encoding. Finally, subsection 5.4 formulates the population sizing model for the link-biased encoding and presents empirical results.

5.1 Functionality of the Link-biased Encoding

In this subsection we want to describe the link-biased encoding and review some of its properties (compare Palmer (1994), Palmer and Kershenbaum (1994a), and Palmer and Kershenbaum (1994b)).

When using the more general LNB encoding, a tree is represented by a string of real-valued weights associated with the nodes and edges of a graph. There are different types of LNB encodings. The link-biased encoding uses weights only associated with the edges of a graph, whereas the node-biased encoding uses weights only associated with the nodes of a graph. Palmer proposed in his original work the LNB encoding which uses both weights. For the purpose of this investigation we want to use the link-biased version only. When using this variant the chromosome b holds the biases (weights) for the links, and has length $n(n-1)/2$ for an n node network.

When constructing the phenotype (the tree) from the genotype (the bias vector b containing the weights), the weight associated to an edge is temporally added to the cost of an edge. To get the represented tree, Prim's algorithm (Prim, 1957) is used to find a minimum spanning tree (MST) from the modified edge costs. By running Prim's algorithm, links with low cost will be used with high probability, whereas edges with high costs will not exist in the tree.

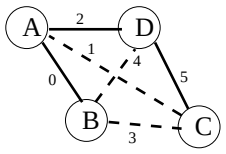


Figure 8: We show an example tree for the link-biased encoding. The numbers indicate the number of a link.

As already mentioned, we only want to use the link-biased encoding for our investigation. Therefore, a chromosome holds weight only for the $n(n-1)/2$ edges. The weights are floating values between zero and one. The original cost d_i of a link i is modified by the elements of the bias vector b_i as

$$d'_i = d_i + P_1 b_i d_{max} \tag{12}$$

where d'_i is the modified cost of a link, d_{max} is the largest cost of a link ($d_{max} = \max(d_i)$), P_1 is the link-specific bias, and $i \in \{0, \ldots, \frac{1}{2}n(n-1)-1\}$ indicates the number of a link. The parameter P_1 controls the influence of the link-specific bias and has a large impact on the structure of the tree. For $P_1 = 0$ the link-specific weights have no influence and only the MST calculated based on d_i can be represented.

The construction of the phenotype can be implemented with a Fibonacci heap and goes with $O(n^2)$. The structure of the represented tree depends not only on the bias values b_i but also on the given cost of the links d_i. The same link-biased individual can represent different trees if different costs for the links are used. Therefore, we assume in our experiments that the cost d_i of the links remains constant and does not change during the run of a GA.

To illustrate the construction of the tree from the bias vector we want to give a brief example. We use the link-biased encoding and for representing a tree with $n = 4$ nodes the genotype is of length $l = n(n-1)/2 = 6$. For the example we want to use the link-biased individual $b = \{0.1, 0.6, 0.2, 0.1, 0.9, 0.3\}$. With $P_1 = 1$ and using the link costs $d = \{10, 30, 20, 40, 10, 20\}$ we can calculate the modified cost according to equation 12 as $d' = \{14, 54, 28, 44, 56, 32\}$. Notice that $d_{max} = 40$. The represented tree, that is calculated as the MST tree using the modified link costs d', is shown in Figure 8. The six possible edges are labeled from 0 to 5 and the tree consists of the edges between A and B (link 0 with $d'_0 = 14$), A and D (link 2 with $d'_2 = 28$), and C and D (link 5 with $d'_5 = 32$).

For further information about the encoding the reader is referred to Palmer (1994).

5.2 Redundancy of the Link-biased Encoding

The following subsection investigates the redundancy of the link-biased encoding.

Representations that assign a discrete, non-infinite number of different phenotypes to genotypes that consist of real values are redundant. Each phenotype can be represented by an infinite number of different genotypes. Consequently, the link-biased representation is a redundant representation. Furthermore, the link-biased encoding is synonymously redundant. Genotypes that represent the same phenotype are next to each other in the mutational space. Small mutations of the weights b_i often do not change the represented phenotype, or only slightly by one edge. Even large mutations that strongly change one weight b_i result only in a change of up to two edges. As a result of the synonymous redundancy of the link-biased encoding, the models from section 3 can be used to predict the influence of the redundant link-biased encoding on the performance of GAs.

In the previous subsection we already noted that for the link-biased encoding the mapping from the genotypes to the phenotypes depends on the link-specific bias P_1. Therefore, to be able to predict the expected GA performance when using the link-biased encoding, it must be investigated how P_1 influences the characteristics of the encoding. The following investigation is an example performed for the link-biased encoding, but the investigation approach is general and can be transferred to any other representation. Factors needing to be examined are:

- Size of the search space.
- Synonymy of the redundant representation.
- Order of redundancy.
- Over- and underrepresentation.

We discuss these aspects in the following paragraphs. When using a (redundant) representation it is important that all possible phenotypes can be represented. A representation should assign at least one genotype to all phenotypes of interest. Otherwise, if no genotype is assigned to some phenotypes, the search space is reduced by the encoding and some possible solutions can never be found by the used search method. The influence of this effect on the performance of a GA is twofold. If the number of accessible solutions is reduced but the optimal solution is still accessible, GA performance increases. On the other hand, if the optimal solution is no longer accessible, all search methods must fail. Therefore, a reduction of the phenotypic search space should be avoided if no problem-specific knowledge about the optimal solution exists. When using the link-biased encoding, the number of accessible solutions depends on P_1 (compare Gaube and Rothlauf (2001) and Rothlauf (2002)). If P_1 is very large, all possible phenotypes can be represented using this encoding. At the other extreme, for P_1 very small ($P_1 \to 0$), only the MST calculated from the link costs d_i can be represented. As long as $P_1 \gtrsim 1$ every possible phenotype can be encoded as the additional overall bias $P_1 b_i d_{max}$ (compare equation 12) can always be larger than any of the original cost of the link d_i. If $P_1 \lesssim 1$ some of the possible trees can not be encoded using the link-biased encoding.

In our proposed model describing the influence of redundant representations on GA performance we assumed that non-uniform redundancy changes the initial supply. If we want to use this model for predicting GA performance we must ensure that the considered representation is synonymously redundant. If a representation is not synonymously redundant, the standard search operators no longer work properly and GEAs fail (compare subsection 2.2). The link-biased encoding is synonymously redundant independently of the parameter P_1. Even if the number of accessible solutions decreases with lower values of P_1 a mutation operator always results in the same, or a slightly different, phenotype.

When using different types of redundant representations, it is important if the order k_r of redundancy changes. In subsection 3.2 we saw that the population size N goes with $O(2^{k_r})$ for

synonymously redundant representations. Therefore, k_r has a strong influence on GA performance. For the real-valued link-biased encoding we can assume that k_r remains independent of P_1.

Finally, when using a redundant representation it must be investigated whether some phenotypes are over- or underrepresented. Subsection 3.2 has shown that the necessary population size n goes with $O(1/r)$. In general, the parameter r is problem-specific and depends on the specific instance of a problem. GA performance remains unchanged if a synonymously redundant representation is uniformly redundant. If a representation is non-uniformly redundant, some instances of a problem will become easier for the search method (those where the optimal solution is overrepresented) and some instances will become more difficult (those where the optimal solution is underrepresented). For the link-biased encoding we have already seen that solutions that are similar to the MST are increasingly overrepresented with decreasing P_1. For very small P_1 only a tiny fraction of genotypes represent a solution different from the MST. We have shown in previous work (Rothlauf, 2002, subsection 6.3.5) that only for large values of P_1 ($P_1 \to \infty$) the representation is approximately uniformly redundant. As a result, there is a continuum between uniform redundancy (P_1 very large) and complete non-uniform redundancy ($P_1 = 0$), which can be controlled by the parameter P_1.

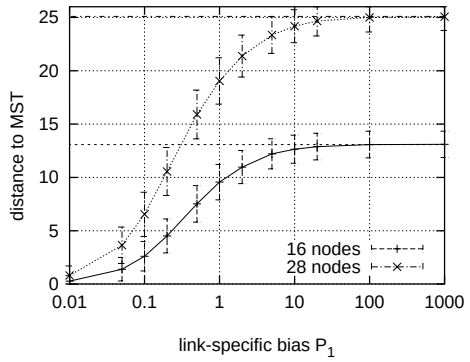


Figure 9: Phenotypic distance of a randomly generated link-biased individual to the minimum spanning tree for trees with $n = 16$ and $n = 26$ nodes. The distance between the two individuals indicates the number of links that are different.

In the remaining paragraphs of this subsection we investigate this continuum and examine how the overrepresentation of specific edges can be controlled by the representation-specific parameter P_1. We want to start with an investigation into how similar a randomly created individual is compared to the MST. The similarity between two trees (the MST and the randomly created individual) is measured by calculating the distance between both trees. The distance between two trees measures the number of different edges. In Figure 9 we show the phenotypic distance of a randomly created link-biased individual to the MST for $n = 16$ and $n = 28$. The error bars show the standard deviations. The dotted lines indicate the distance of a randomly created individual towards the MST when using a non-redundant representation (for example Prüfer numbers (Prüfer, 1918)). The results show that for large values of P_1 a randomly created link-biased individual has about the same distance towards the MST as a non-redundant encoding. Therefore, it can be assumed that with P_1 large enough the link-biased encoding is uniformly redundant. With decreasing values of P_1 the represented trees become more and more MST-like and the link-biased encoding becomes more and more non-uniformly redundant.

The analysis of the distance of a randomly generated link-biased individual towards the MST reveals that the overrepresentation of a specific solution (the MST) strongly depends on the link-

specific bias P_1. To be able to calculate the overrepresentation of specific edges (we need this for the population sizing model we derive in subsection 5.4) we want to examine how the probability P_r that an edge contained in a randomly created link-biased individual is also contained in the MST depends on P_1. For non-redundant or uniformly redundant representations the probability P_r^u can be calculated as

$$P_r^u = \frac{2n}{n(n-1)}. \tag{13}$$

Table 5 compares for different problem sizes n the probability P_r^u for non-redundant representations to empirical results for P_r when using a large link-specific bias ($P_1 = 1\ 000\ 000$). It can be seen that for large values of P_1 the probability P_r (that an edge contained in a randomly created link-biased individual is also contained in the MST) equals the probability P_r^u (that a randomly chosen edge is part of the MST). Therefore, for large values of P_1 the encoding becomes uniformly redundant.

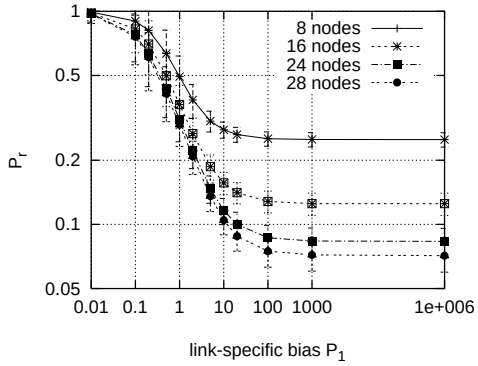


Figure 10: Probability that a link of a randomly generated tree is part of the MST over the link-specific bias P_1.

n	P_r^u	P_r for $P_1 = 1\ 000\ 000$	
		mean	σ
8	0.25	0.2497	0.01292
12	0.1667	0.1665	0.01655
16	0.125	0.1250	0.01495
20	0.1	0.0998	0.09982
24	0.0834	0.0832	0.01239
28	0.0714	0.0713	0.01163

Table 5: A comparison between P_r^u and P_r when using $P_1 = 1\ 000\ 000$.

Consequently, Figure 10 plots for the link-biased encoding how P_r depends on the link-specific bias P_1. The results show the mean and the standard deviation for 8, 16, 24, and 28 node trees. For large values of P_1 ($P_1 > 100$), P_r equals P_r^u and we get the values shown in Table 5. With decreasing P_1 the edges contained in a randomly created individual are more and more often also contained in the MST. For small values of P_1, all edges of a randomly created individual are with high probability P_r also part of the MST.

After discussing how the redundancy of the link-biased encoding and the overrepresentation of specific edges depends on the link-specific bias P_1, in the following subsections we want to formulate a model based on the results from subsection 3.2 which describes how GA performance depends on P_1.

5.3 Population Sizing Model for the One-Max Tree Problem

In this subsection we review a population sizing model which we can use for the uniformly redundant (P_1 must be large) link-biased encoding. The model is valid for the one-max tree problem

and was developed in previous work (Rothlauf et al., 2002). The one-max tree problem and the corresponding population sizing model is used in our experiments in subsection 5.4.

The one-max tree problem (Rothlauf et al., 2002) defines a tree optimization problem where an optimal solution (tree) is chosen either randomly or by hand. The structure of this tree can be determined: It can be the MST, a star, a list, or any other arbitrary tree with n nodes.

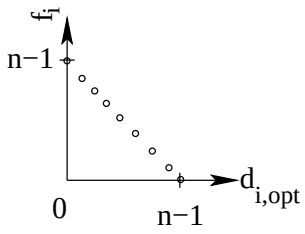


Figure 11: The one-max tree problem

In this problem the fitness of a tree G_i is defined as the number of edges it has in common with the best solution G_{opt}:

$$f_i = n - 1 - d_{i,opt},$$

where the distance $d_{i,opt}$ between two trees G_i and G_{opt} is defined as

$$d_{i,opt} = \frac{1}{2} \sum_{i=1}^{n-1} \sum_{j=0}^{i-1} |l_{ij}^{i} - l_{ij}^{opt}|.$$

l_{ij} is 1 if the link from node i to node j exists and 0 if it does not exist. This definition of distance between two trees is based on the Hamming distance (Hamming, 1980) and $d_{i,opt} \in \{0, 1, \ldots, n-2\}$. In all our experiments G_{opt} is always the minimum spanning tree.

Earlier work (Rothlauf et al., 2002) presented a population sizing model for the one-max tree problem which was derived for the Network Random Key representation. The Network Random Key representation is almost identical to the link-biased encoding using large values of P_1. Both encodings are synonymous uniformly redundant representations defined on real-valued strings of the same length. Only the construction of the tree from the genotypic weights is different. Network Random Keys use Kruskal's algorithm and do not consider the original costs d_i between the links, whereas the link-biased encoding uses Prim's algorithm for the construction of the phenotypes and considers d_i. Therefore, the population sizing model for the Network Random Keys is valid for the link-biased encoding if a large link-specific bias P_1 is used. A large value of P_1 is necessary to ensure uniform redundancy. The model is formulated as

$$N = -\frac{\sqrt{\pi}}{4} \ln(\alpha) \sqrt{n(n-1)(n-2)} \approx -\frac{\sqrt{\pi}}{4} \ln(\alpha) n^{1.5}, \tag{14}$$

where α is the probability of failure and n is the number of nodes. It can be seen that the necessary population size N goes with $O(n^{1.5})$. For further information about the population sizing model the reader is referred to the original work.

5.4 Population Sizing for the Link-Biased Encoding

Subsection 5.2 has shown that with decreasing link-specific bias P_1 the link-biased encoding over-represents solutions similar to the MST. In the following subsection we show that we are able to give good predictions on how GA performance depends on the link-specific parameter P_1 by combining the population sizing model from the previous subsection which only holds for the uniformly redundant link-biased encoding (P_1 must be large) with the population sizing model from subsection 3.2 which explains the influence of non-uniformly redundant representations on GA performance. We finally formulate the population sizing model for the link-biased encoding and present experimental results.

In all following experiments the optimal solution for the one-max problem is always the MST. We want to calculate for a GA using the link-biased encoding the population size N that is necessary for finding the optimal solution (the MST) with some probability P_{opt}. The optimal solution is

correctly found by a GA if all of the $n-1$ links of the optimal solution are correctly identified. Therefore, $P_{opt} = (1-\alpha)^{n-1}$, where α is the probability of error for one link. We get for

$$\alpha = 1 - \exp\left(\frac{\log(P_{opt})}{n-1}\right).$$

Substituting α into equation 14 results in

$$N = -\frac{\sqrt{\pi}}{4} \ln\left(1 - \exp\left(\frac{\log(P_{opt})}{n-1}\right)\right) \sqrt{n(n-1)(n-2)}. \tag{15}$$

This population sizing model should give us good predictions for the expected minimal population size using the link-biased encoding with a large link-specific bias P_1. The large link-bias ensures that the encoding is uniformly biased.

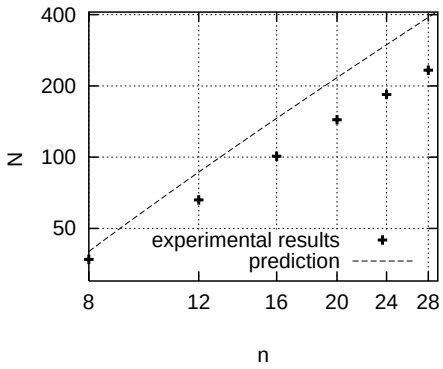


Figure 12: Necessary population size N over the problem size n for the one-max tree problem. The optimal solution is the MST and the link-specific bias is set to $P_1 = 1\ 000\ 000$ to ensure uniform redundancy. The results show that the used population sizing models gives an acceptable approximation of the expected GA performance.

Figure 12 shows the theoretical prediction from above and the experimental results for the link-biased encoding with $P_1 = 1\ 000\ 000$. The plots show the necessary population size N over the problem size n for $P_{opt} = 0.95$. We performed 500 runs for each population size and the resolution for N is 1. As in previous experiments we used no mutation, but only uniform crossover. In all runs we use tournament selection without replacement of size 3 and each run is stopped after the population is fully converged. Because the encoded phenotype depends on the cost d_i between the different nodes, we randomly placed for every run the nodes on a 1000x1000 square. The cost of a link d_i is calculated as the Euclidean distance between the two connected nodes.

Although the population sizing model described in equation 15 slightly overestimates the necessary population size N, it still allows a good approximation of the experimental results. As we are mainly interested in investigating the influence of P_1 on the solution quality, and not on the development of a highly accurate population sizing model, we are satisfied with the accuracy of this population sizing model. It can be seen that the necessary population size N increases with about $O(n^{1.5})$.

In the following we want to consider that the link-biased encoding becomes non-uniformly redundant with decreasing P_1. With lower P_1 the links that are contained in the MST are over-represented by the encoding. Therefore, GA performance increases and the population size that is necessary to find the optimal solution (the MST) decreases. We have seen in subsection 3.2 that the

necessary population size N goes with $O(2^{k_r}/r)$. r is the number of genotypic BBs that represent the optimal phenotypic BB. For the one-max problem we can assume that the size of the BBs k equals one and that each possible link is one phenotypic BB.

We have to determine how the different phenotypic BBs (the possible edges in the tree) are overrepresented by the link-biased representation. In subsection 5.2 we have introduced the probability P_r that a link contained in a randomly created individual is also part of the optimal solution. We can assume that the probability P_r is proportional to r ($P_r = const * r$). Doubling the probability P_r means that a specific link of a randomly created individual is twice as often also contained in the optimal solution (the MST). Therefore, doubling P_r has the same effect as doubling r. Furthermore, we can assume that the character of the link-biased encoding does not change for different values of P_1 and k_r remains constant. Therefore, the population size N when using a non-uniformly redundant link-biased encoding goes with $O(1/P_r)$. Using equation 15 we finally get

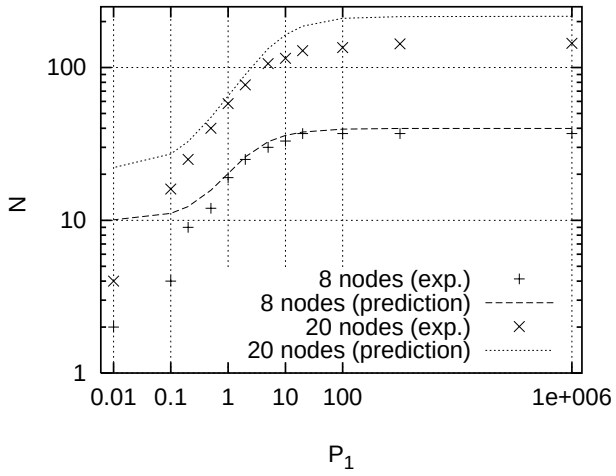
$$N = -\frac{P_r^u}{P_r}\frac{\sqrt{\pi}}{4}\ln\left(1-\exp\left(\frac{\log(P_{opt})}{n-1}\right)\right)\sqrt{n(n-1)(n-2)},$$

where P_r^u indicates P_r for $P_1 \to \infty$ (compare equation 13). The values of P_r depend on the link-specific bias P_1 and are shown in Figure 10 for different problem sizes n.

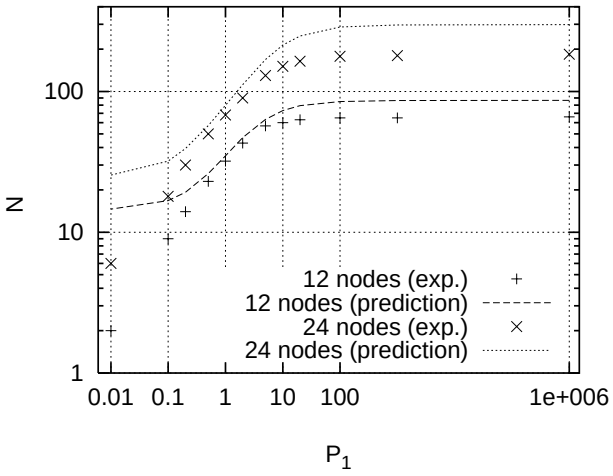
The theoretical predictions and the empirical results for different problem sizes are shown in Figure 13(a) (8 and 20 node one-max problem), Figure 13(b) (12 and 24 node one-max problem), and Figure 13(c) (16 and 28 node one-max tree). The results are split into three plots due to illustrative purposes. The plots show how the necessary population size N depends on the link-specific bias P_1. The probability of finding the optimal solution (the MST) is $P_{opt} = 0.95$. For determining the relationship between P_1 and P_r which we discussed in subsection 5.2, we used the results plotted in Figure 10. The lines show the theoretical predictions from our population sizing model and the points show the experimental results. In all runs the optimal solution was the MST and we used the same parameters as for the uniformly redundant link-biased encoding, whose details are described above.

The results show that the proposed population sizing model gives us a good prediction on how the performance of a GA depends on the link-specific bias P_1. There is only a small difference between the predicted value for N and the actual experimental results. As expected, the population size N declines with decreasing P_1 and the problem becomes easier to solve for a GA. Furthermore, we can see that for small values of $P_1 < 1$ the necessary population size N strongly declines and the experimental population size drops much faster than predicted. This is because for $P_1 < 1$ (compare subsection 5.2) the link-biased encoding does not allow us to encode all possible trees and the search space collapses. Only trees that are similar to the MST can be encoded. Small values of P_1 result in high values of P_r (compare Figure 10) what means that most of the links of a randomly created individual are also part of the optimal solution (the MST). At the extreme, for $P_1 \to 0$ ($P_r \to 1$), the link-biased encoding can only encode the optimal solution (the MST) and the necessary population size $N \to 0$.

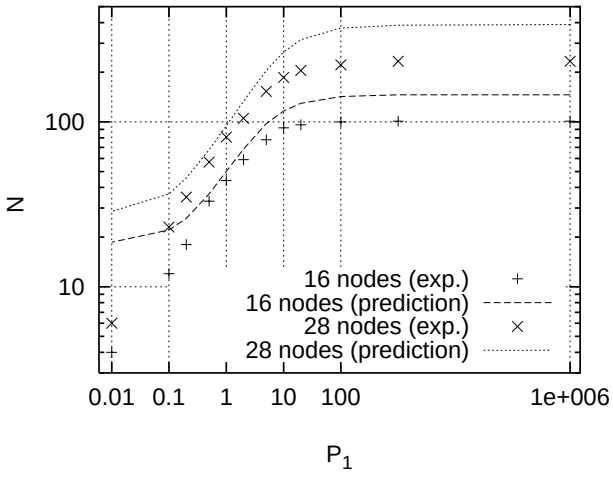
This section illustrated that the proposed theoretical concepts describing the influence of synonymously redundant representations on the performance of GAs can not only be used for binary representations but also for real-valued representations like the link-biased encoding. The link-biased encoding was chosen as it allows us to adjust the level of overrepresentation in a systematic way. The presented results have shown that the proposed theory from section 3 predicts the expected GA behavior well.



(a) 8 and 20 node one-max tree



(b) 12 and 24 node one-max tree



(c) 16 and 28 node one-max tree

Figure 13: We show how the population size N which is necessary for finding the optimal solution with probability $P_{opt} = 0.95$ depends on the link-specific bias P_1. In all runs the optimal solution was the MST. The results show that the proposed population sizing model gives good predictions for the expected solution quality. For small values of P_1 the populations size N strongly decreases as the size of the search space collapses and only the optimal solution (the MST) can be represented.

6 Future Research

Based on this study some topics require further investigation.

In this work we proposed a model describing the effect of synonymously redundant representations on the performance of GAs. We illustrated the validity of our approach for the binary trivial voting mapping and the real-valued link-biased encoding. An interesting direction for further research is to apply this model to other problems and other problem domains. We believe that using this approach could help to gain a better theoretical understanding of the properties of redundant representations. Furthermore, doing this we will be able to find possible limitations of the proposed approach. In this work we only modeled the influence of synonymously redundant representations as an effect of initial supply. We have to investigate under which circumstances other, until now neglected effects, like improper mixing of BBs, can have an influence on GA performance.

We already discussed in subsection 2.2 that the use of non-synonymously redundant representations results in low GA performance as the genetic operators no longer work properly. When using non-synonymously redundant representations genetic operators like crossover can create lower quality BBs even if the parents only consist of high quality BBs. An example of non-synonymous redundancy is that $x_p = 0$ is represented by the genotypes $\{00, 11\}$, and $x_p = 1$ is represented by $\{01, 10\}$. Combining the genotypes 01 and 10 can result in the offspring 11 or 00 which both represent a different phenotype. We believe that non-synonymously redundant representations can be modeled by combining results for low-locality representations with the results for synonymously redundant representations we presented here. A profound model describing non-synonymously redundant representations would allow us to analyze representations in a theory-guided matter and to solve some of the disputed topics regarding the benefits of redundant representations we mentioned in subsection 2.1. Some ideas concerning this topic can also be found in Rothlauf (2002).

In this work we only considered crossover-based search and neglected the influence of redundant representations on mutation-based search approaches. However, we believe that many of the discussed topics are also relevant when using mutation. Following subsection 2.2 we believe that, in analogy with the results from Knowles and Watson (2002), using non-synonymously redundant representations will reduce the performance of mutation-based search. As these representations have low locality, mutation will not work properly and the search becomes random. Furthermore, there is some theoretical evidence (Whitley, Rana, & Heckendorn, 1997; Rana & Whitley, 1997; Whitley, 1999; Whitley, 2000) that mutation-based search only performs well if the connectivity of the phenotypic search space is preserved by the used representation. If the connectivity is either not preserved, such as for low locality representations, or greatly increased (what results in a reduction of the relevant connectivity) like in many non-synonymously redundant representations, the performance of mutation-based search decreases. In contrast, we expect when using synonymously redundant representations that mutation-based search shows similar behavior and performance as when using crossover-based search. Using synonymously redundant representations introduces many plateaus in the fitness landscape but does not change the structure of the search space. Mutation can still easily find neighboring phenotypes. When using non-uniformly redundant representations some plateaus in the fitness landscape are increased which increases the probability that mutation finds the solution represented by the genotypes forming this plateau. As a result, the performance of mutation-based search increases if a synonymously redundant representation overrepresents the optimal solution and decreases otherwise.

Finally, we want to emphasize the importance of the proposed concepts for the field of genetic programming (GP). Most of the representations used in GP are redundant. When encoding program structures by using either some type of tree encoding or some binary encoding like in grammatical evolution (O'Neill & Ryan, 2001), each phenotype is represented by more than one

genotype. Therefore, these representations are redundant and the proposed classification into synonymously and non-synonymously redundant representations can be applied. Furthermore, it must be examined which of the used representations are uniformly and which are non-uniformly redundant. Our belief is that many of the representations used in GP are non-uniformly redundant and overrepresent some phenotypes. We expect that by applying the proposed theory to these types of representations we can help to build more efficient GP systems.

7 Conclusions

This paper started with a short review of the literature concerning the use of redundant representations in evolutionary computation. This was followed by the development of a classification of redundant representations which distinguished between synonymously and non-synonymously redundant representations. Furthermore, it discussed how the synonymy of a representation influences genetic search. Then, it developed a population sizing model for synonymously redundant representations based on the assumption that a representation affects the initial supply. Representations were characterized by two parameters and a theoretical model was developed that predicted the population size and the number of generations that were necessary for solving a problem. Furthermore, the models for predicting the performance of GAs were used for the synonymously redundant trivial voting mapping and variants of it. Results for the one-max problem and concatenated traps have were shown and theoretical predictions were compared to empirical results. The paper also illustrated how the concepts of redundant representations can be used for the link-biased encoding. The link-biased encoding represents trees by using real-valued vectors and is synonymously, non-uniformly redundant. When using this encoding the overrepresentation of some phenotypes can be controlled by the link-specific bias P_1. The paper described the functionality of the encoding, investigated how the overrepresentation of some phenotypes depends on the link-specific bias, formulated a population sizing model, and presented empirical results on how GA performance depends on the link-specific bias. Finally, some directions of further research were presented.

This paper investigated how redundant representations influence the performance of crossover-based GAs. It distinguished between between synonymously and non-synonymously redundant representations and illustrated that non-synonymously redundancy does not allow genetic operators to work properly and therefore reduces the efficiency of evolutionary search. When using synonymously redundant representations, GA performance depends on the change of the initial supply. Based on this observation models were developed that give the necessary population size for solving a problem, and the number of generations as $O(2^{k_r}/r)$, where k_r is the order of redundancy and r is the number of genotypic BBs that represent the optimal phenotypic BB. As a result, uniformly redundant representations do not change the behavior of genetic algorithms. Only by increasing r, which means overrepresenting the optimal solution, does GA performance increase. By contradistinction, GA performance decreases if the optimal solution is underrepresented. Therefore, non-uniformly redundant representations can only be used advantageously if there exists a-priori some information about the optimal solution.

The validity of the proposed theoretical concepts is illustrated for two different examples. Firstly, the influence of different variants of the redundant trivial voting mapping on GA performance is investigated. The results show that the developed population sizing and time to convergence models allow an accurate prediction of the expected solution quality and time. Secondly, the results for the link-biased representation show that the proposed theoretical concepts are not only valid for binary representations, but can also be used for real-valued representations.

In analogy to the results for the trivial voting mapping, the population sizing model gives accurate predictions on how the expected solution quality depends on the overrepresentation of the optimal solution.

Based on the presented results we strongly encourage users and researchers in evolutionary computation to use the developed concepts. The proposed classification, population sizing, and time to convergence models allow us to evaluate redundant representations in a systematic and theory-guided matter. This approach will help users and researchers to answer some of the disputed questions regarding the benefits of redundant representations and to use redundant representations such that they increase the performance, reliability and efficiency of evolutionary computation methods.

References

Abramowitz, M., & Stegun, I. A. (1972). *Handbook of mathematical functions*. New York: Dover Publications.

Abuali, F. N., Wainwright, R. L., & Schoenefeld, D. A. (1995). Determinant factorization: A new encoding scheme for spanning trees applied to the probabilistic minimum spanning tree problem. In Eschelman, L. (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 470–477). San Francisco, CA: Morgan Kaufmann.

Ackley, D. H. (1987). *A connectionist machine for genetic hill climbing*. Boston: Kluwer Academic.

Banzhaf, W. (1994). Genotype-phenotype-mapping and neutral variation – A case study in genetic programming. In Davidor, Y., Schwefel, H.-P., & Männer, R. (Eds.), *Parallel Problem Solving from Nature- PPSN III* (pp. 322–332). Berlin: Springer.

Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.) (1999). *Proceedings of the Genetic and Evolutionary Computation Conference: Volume 1*. San Francisco, CA: Morgan Kaufmann Publishers.

Barnett, L. (1997). *Tangled webs: Evolutionary dynamics on fitness landscapes with neutrality*. Master's thesis, School of Cognitive Sciences, University of East Sussex, Brighton.

Barnett, L. (1998, June 27–29). Ruggedness and neutrality: The NKp family of fitness landscapes. In Adami, C., Belew, R. K., Kitano, H., & Taylor, C. (Eds.), *Proceedings of the 6th International Conference on Artificial Life (ALIFE-98)* (pp. 18–27). Cambridge, MA, USA: MIT Press.

Barnett, L. (2001). Netcrawling - optimal evolutionary search with neutral networks. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC01* (pp. 30–37). Piscataway, NJ: IEEE Press.

Cohoon, J. P., Hegde, S. U., Martin, W. N., & Richards, D. (1988). Floorplan design using distributed genetic algorithms. In *IEEE International Conference on Computer Aided-Design* (pp. 452–455). IEEE.

Darwin, C. (1859). *On the origin of species*. London: John Murray.

Dasgupta, D. (1995). Incorporating redundancy and gene activation mechanisms in genetic search for adapting to non-stationary environments. In Chambers, L. (Ed.), *Practical Handbook of Genetic Algorithms* (Chapter 13, pp. 303–316). CRC Press.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 61–69). San Mateo, CA: Morgan Kaufmann.

Deb, K., & Goldberg, D. E. (1993). Analyzing deception in trap functions. In Whitley, L. D. (Ed.), *Foundations of Genetic Algorithms 2* (pp. 93–108). San Mateo, CA: Morgan Kaufmann.

Ebner, M., Langguth, P., Albert, J., Shackleton, M., & Shipman, R. (2001, 27-30 May). On neutral networks and evolvability. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001* (pp. 1–8). COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea: IEEE Press.

Eshelman, L. J., & Schaffer, J. D. (1991). Preventing premature convergence in genetic algorithms by preventing incest. In Belew, R. K., & Booker, L. B. (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 115–122). San Mateo, CA: Morgan Kaufmann.

Gaube, T., & Rothlauf, F. (2001). The link and node biased encoding revisited: Bias and adjustment of parameters. In Boers, E. J. W., Cagnoni, S., Gottlieb, J., Hart, E., Lanzi, P. L., Raidl, G. R., Smith, R. E., & Tijink, H. (Eds.), *Applications of evolutionary Computing: Proc. EvoWorkshops 2001* (pp. 1–10). Berlin: Springer.

Gerrits, M., & Hogeweg, P. (1991). Redundant coding of an NP-complete problem allows effective Genetic Algorithm search. In Schwefel, H.-P., & Männer, R. (Eds.), *Parallel Problem Solving from Nature* (pp. 70–74). Berlin: Springer-Verlag.

Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, *6*, 333–362.

Gottlieb, J., Julstrom, B. A., Raidl, G. R., & Rothlauf, F. (2001). Prüfer numbers: A poor representation of spanning trees for evolutionary search. In Spector, L., E., G., Wu, A., B., L. W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., & Burke, E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference 2001* (pp. 343–350). San Francisco, CA: Morgan Kaufmann Publishers.

Hamming, R. (1980). *Coding and information theory.* Prentice-Hall.

Harik, G. R., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. In Bäck, T. (Ed.), *Proceedings of the Forth International Conference on Evolutionary Computation* (pp. 7–12). New York: IEEE Press.

Harvey, L., & Thompson, A. (1997). Through the labyrinth evolution finds a way: A silicon ridge. *Lecture Notes in Computer Science*, *1259*, 406–422.

Huynen, M. (1996). Exploring phenotype space through neutral evolution. *J. Mol. Evol.*, *43*, 165–169.

Huynen, M., Stadler, P., & Fontana, W. (1996). Smoothness within ruggedness: The role of neutrality in adaptation. In *Proceedings of the National Academy of Science USA, 93* (pp. 397–401).

Julstrom, B. A. (1999). Redundant genetic encodings may not be harmful. See Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, and Smith (1999), pp. 791.

Kargupta, H. (2000). The genetic code-like transformations and their effect on learning functions. See Schoenauer, Deb, Rudolph, Yao, Lutton, Merelo, and Schwefel (2000), pp. 99–108.

Kargupta, H. (2001). A striking property of genetic code-like transformations. *Complex Systems*, *13*(1), 1–32.

Kimura, M. (1983). *The neutral theory of molecular evolution*. Cambridge University Press.

Knowles, J. D., & Watson, R. A. (2002). On the utility of redundant encodings in mutation-based evolutionary search. In Merelo, J. J., Adamidis, P., Beyer, H.-G., Fernandez-Villacanas, J.-L., & Schwefel, H.-P. (Eds.), *Parallel Problem Solving from Nature, PPSN VII* (pp. 88–98). Berlin: Springer-Verlag.

Krishnamoorthy, M., & Ernst, A. T. (2001). Comparison of algorithms for the degree constrained minimum spanning tree. *Journal of Heuristics*, *7*, 587–611.

Miller, B. L., & Goldberg, D. E. (1996a). Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, *4*(2), 113–131.

Miller, B. L., & Goldberg, D. E. (1996b). Optimal sampling for genetic algorithms. In Dagli, C. H., Akay, M., Chen, C. L. P., Fernández, B. R., & Ghosh, J. (Eds.), *Proceedings of the Artificial Neural Networks in Engineering (ANNIE '96) conference*, Volume 6 (pp. 291–297). New York: ASME Press.

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49.

O'Neill, M., & Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, *5*(4), 349–358.

Palmer, C. C. (1994). *An approach to a problem in network design using genetic algorithms*. unpublished PhD thesis, Polytechnic University, Troy, NY.

Palmer, C. C., & Kershenbaum, A. (1994a). Representing trees in genetic algorithms. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, Volume 1 (pp. 379–384). Piscataway, NJ: IEEE Service Center.

Palmer, C. C., & Kershenbaum, A. (1994b). *Two algorithms for finding optimal communication spanning trees*. IBM research report RC-19394.

Prim, R. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, *36*, 1389–1401.

Prüfer, H. (1918). Neuer Beweis eines Satzes über Permutationen. *Archiv für Mathematik und Physik*, *27*, 742–744.

Raidl, G. R., & Julstrom, B. A. (2000). A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In Carroll, J., Damiani, E., Haddad, H., & Oppenheim, D. (Eds.), *Proceedings of the 2000 ACM Symposium on Applied Computing* (pp. 440–445). ACM Press.

Rana, S. B., & Whitley, L. D. (1997). Bit representations with a twist. In Bäck, T. (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms* (pp. 188–195). San Francisco: Morgan Kaufmann.

Reidys, C. M., & Stadler, P. F. (1998). Neutrality in fitness landscapes. *Applied Mathematics and Computation*, *117*(2–3), 321–350.

Ronald, S., Asenstorfer, J., & Vincent, M. (1995). Representational redundancy in evolutionary algorithms. In *1995 IEEE International Conference on Evolutionary Computation*, Volume 2 (pp. 631–636). Piscataway, NJ: IEEE Service Center.

Rothlauf, F. (2002). *Representations for genetic and evolutionary algorithms*. Studies on Soft Computing and Fuzziness. Berlin: Springer Verlag.

Rothlauf, F., & Goldberg, D. E. (2000). Prüfernumbers and genetic algorithms: A lesson on how the low locality of an encoding can harm the performance of GAs. See Schoenauer, Deb, Rudolph, Yao, Lutton, Merelo, and Schwefel (2000), pp. 395–404.

Rothlauf, F., Goldberg, D. E., & Heinzl, A. (2002). Network random keys – A tree network representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, *10*(1), 75–97.

Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., & Schwefel, H.-P. (Eds.) (2000). *Parallel Problem Solving from Nature, PPSN VI*. Berlin: Springer-Verlag.

Schuster, P. (1997). Genotypes with phenotypes: Adventures in an RNA toy world. *Biophys. Chem.*, *66*, 75–110.

Shackleton, M., Shipman, R., & Ebner, M. (2000, 6-9 July). An investigation of redundant genotype-phenotype mappings and their role in evolutionary search. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00* (pp. 493–500). La Jolla Marriott Hotel La Jolla, California, USA: IEEE Press.

Shipman, R. (1999). Genetic redundancy: Desirable or problematic for evolutionary adaptation? In *Proceedings of the 4th International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA)* (pp. 1–11). Springer Verlag.

Shipman, R., Shackleton, M., Ebner, M., & Watson, R. (2000). Neutral search spaces for artificial evolution: A lesson from life. In Bedau, M., McCaskill, J., Packard, N., & Rasmussen, S. (Eds.), *Proceedings of Artificial Life VII* (pp. section III (Evolutionary and Adaptive Dynamics)). MIT Press.

Shipman, R., Shackleton, M., & Harvey, L. (2000). The use of neutral genotype-phenotype mappings for improved evoutionary search. *British Telecom Technology Journal*, *18*(4), 103–111.

Smith, T., Husbands, P., & M., O. (2001a). *Evolvability, neutrality and search space* (Technical Report 535). School of Cognitive and Computing Sciences, University of Sussex.

Smith, T., Husbands, P., & M., O. (2001b). Neutral networks and evolvability with complex genotype-phenotype mapping. In *Proceedings of the European Converence on Artificial Life: ECAL2001* (pp. 272–281).

Smith, T., Husbands, P., & M., O. (2001c). Neutral networks in an evolutionary robotics search space. In of Electrical, I., & Engineers, E. (Eds.), *Proceedings of 2001 IEEE International Conference on Evolutionary Computation* (pp. 136–145). Piscataway, NJ: IEEE Service Center.

Thierens, D., & Goldberg, D. E. (1993). Mixing in genetic algorithms. In Forrest, S. (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 38–45). San Mateo, CA: Morgan Kaufmann.

Toussaint, M., & Igel, C. (2002). Neutrality: A necessity for self-adaptation. In Fogel, D. B., El-Sharkawi, M. A., Yao, X., Greenwood, G., Iba, H., Marrow, P., & Shackleton, M. (Eds.), *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002* (pp. 1354–1359). IEEE Press.

Whitley, D. (1999). A free lunch proof for gray versus binary encodings. See Banzhaf, Daida, Eiben, Garzon, Honavar, Jakiela, and Smith (1999), pp. 726–733.

Whitley, D. (2000). Functions as permutations: Implications for no free lunch, walsh analysis and statistics. See Schoenauer, Deb, Rudolph, Yao, Lutton, Merelo, and Schwefel (2000), pp. 169–178.

Whitley, D., Rana, S., & Heckendorn, R. (1997). Representation issues in neighborhood search and evolutionary algorithms. In *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science* (Chapter 3, pp. 39–58). West Sussex, England: John Wiley & Sons Ltd.

Yu, T., & Miller, J. (2001). Neutrality and evolvability of Boolean function landscapes. In *Proceedings of the 4th European Conference on Genetic Programming (EuroGP)*, Volume LNCS 2038 (pp. 204–217). Springer.

Yu, T., & Miller, J. (2002). Finding needles in haystacks is not hard with neutrality. In *Proceedings of the 5th European Conference on Genetic Programming (EuroGP)*, Volume LNCS (pp. 13–25). Springer.