The Influence of Binary Representations of Integers on the Performance of Selectorecombinative Genetic Algorithms

Franz Rothlauf

Working Paper 1/2002
February 2002

Working Papers in Information Systems
Editor: Prof. Dr. Armin Heinzl

University of Bayreuth
Department of Information Systems
Universitätsstrasse 30
D-95440 Bayreuth, Germany
Phone +49 921 552807, Fax +49 921 552216
E-Mail: wi@uni-bayreuth.de
Internet: *h*ttp://wi.oec.uni-bayreuth.de

The Influence of Binary Representations of Integers on the Performance of Selectorecombinative Genetic Algorithms

Franz Rothlauf*
Dept. of Information Systems
University of Bayreuth
Universitätsstr. 30
D-95440 Bayreuth/Germany
rothlauf@uni-bayreuth.de

February 18, 2002

Abstract

When using representations for genetic algorithms (GAs) every optimization problem can be separated into a genotype-phenotype and a phenotype-fitness mapping. The genotype-phenotype mapping is the used representation and the phenotype-fitness mapping is the problem that should be solved.

This paper investigates how the use of different binary representations of integers influences the performance of selectorecombinative GAs using only crossover and no mutation. It is shown that the used representation strongly influences the performance of GAs. The binary and gray encoding are two different, well known possibilities to assign bitstring genotypes to integer phenotypes. Focusing our investigation on these two encodings reveals that for the easy integer one-max problem selectorecombinative GAs perform better using binary encoding than using gray encoding. This is surprising as binary encoding is affected with problems due to the Hamming cliff and because there is evidence that shows the superiority of gray encoding. However, the performance of selectorecombinative GAs is determined by the structure of the building blocks (BBs) and not by the structure of the search space. Therefore, the performance difference between the encodings can be explained by analyzing the fitness of the resulting schemata. It reveals for the easy integer one-max problem that binary encoding results in BBs of lower order than gray encoding. Therefore, the integer one-max problem is more difficult and the performance of selectorecombinative GAs is lower when using gray encoding.

1 Introduction

Integer optimization problems are important in many real-world applications. We know from previous work (Liepins & Vose, 1990) that the choice of a proper representation (genotype-phenotype mapping) is crucial for the performance of genetic and evolutionary algorithms (GEAs). When solving integer problems, binary representations of integers like gray or binary encoding are often used.

In this work we want to investigate how binary representations of integers influence the performance of selectorecombinative genetic algorithms (GAs) which only use crossover and selection

* Also with Illinois Laboratory of Genetic Algorithms, University of Illinois at Urbana-Champaign, USA.

and no mutation. The results show large differences in GA performance using different binary representations. Furthermore, we see that selectorecombinative GAs perform better using binary encoding than using gray encoding. This behavior of selectorecombinative GAs can be explained by analyzing the fitness of the resulting schemata.

The paper is structured as follows. In the following section we provide the basis and requisites for our investigations. Section 3 examines how different types of binary representations of integers influence the performance of selectorecombinative GAs. We calculate the number of possible representations and present empirical results. In section 4 we focus on the influence of binary and gray encoding on GA performance. To explain the experimental results presented in subsection 4.1 we analyze in subsection 4.2 the fitness of the resulting schemata. The paper ends with concluding remarks.

2 Binary Representations for Integer Optimization Problems

When using a representation we can separate an optimization problem into a genotype-phenotype mapping and a phenotype-fitness mapping. Consequently, we present in subsection 2.2 the integer optimization problem we want to solve, and in subsection 2.3 binary representations of integers.

2.1 Separating Representations from Optimization Problems

The following subsection provides some basic definitions for our discussion of representations. When using some kind of representation, every optimization problem can be decomposed into a genotype-phenotype mapping f_g, and a phenotype-fitness mapping f_p (Mendel, 1866; Liepins & Vose, 1990).

We define Φ_g as the genotypic search space where the genetic operators such as recombination or mutation are applied to. An optimization problem on Φ_g could be formulated as follows: The search space Φ_g is either discrete or continuous, and the function

$$f(\boldsymbol{x}) : \Phi_g \to \mathbb{R}$$

assigns an element in $\mathbb{R}$ to every element in the genotype space Φ_g. The optimization problem is defined by finding the optimal solution

$$\hat{\boldsymbol{x}} = \max_{\boldsymbol{x} \in \Phi_g} f(\boldsymbol{x}),$$

where $\boldsymbol{x}$ is a vector of decision variables (or alleles), and $f(\boldsymbol{x})$ is the fitness function. The vector $\hat{\boldsymbol{x}}$ is the global maximum.

When using a representation we have to introduce – in analogy to nature – phenotypes and genotypes (Mendel, 1866; Lewontin, 1974). Thus, the fitness function f can be decomposed into two parts. The first maps the genotypic space Φ_g to the phenotypic space Φ_p, and the second maps Φ_p to the fitness space $\mathbb{R}$. Using the phenotypic space Φ_p we get:

$$f_g(\boldsymbol{x}_g) : \Phi_g \to \Phi_p,$$
$$f_p(\boldsymbol{x}_p) : \Phi_p \to \mathbb{R},$$

where $f = f_p \circ f_g = f_p(f_g(\boldsymbol{x}_g))$. The genotype-phenotype mapping f_g is the used representation. f_p represents the fitness function and assigns a fitness value $f_p(\boldsymbol{x}_p)$ to every individual $\boldsymbol{x}_p \in \Phi_p$. The genetic operators are applied to the individuals in Φ_g that means on the level of genotypes (Bagley, 1967; Vose, 1993).

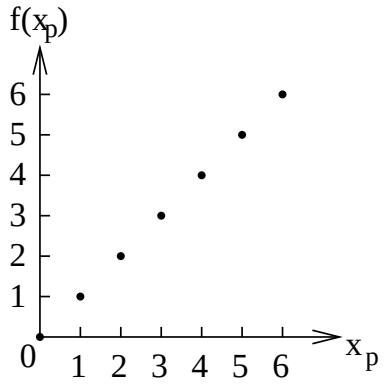


Figure 1: Integer one-max problem

2.2 An Integer Optimization Problem

This subsection defines the integer optimization problem we want to use for our investigations. The problem should be easy and it should be defined on the phenotypes independently of the used representation.

We assume that the fitness function f_p assigns a real number to every individual $x_p \in \mathbb{N}$. Therefore, we get for f_p:

$$f_p(x_p) : \mathbb{N} \to \mathbb{R}.$$

For the optimization problem itself we want to define an integer-specific variation of the one-max problem as

$$f_p(x_p) = x_p. \tag{1}$$

This integer one-max problem is easy for GEAs which use mutation as the main search operator. However, also GEAs using crossover as the main search operator can solve this problem easily with proper population size. An example for the integer one-max problem is given in Figure 1.

2.3 Binary Representations of Integers

In this subsection we present the binary and gray encoding as examples for binary representations of integers.

If we encode integer phenotypes using binary genotypes we have to ask why we do not use integer genotypes for encoding integer phenotypes. In general, instead of using binary strings with cardinality $\chi = 2$, higher χ-ary alphabets could be used for the genotypes. When using a χ-ary alphabet for the genotypic alleles, we are able to encode with one allele χ different phenotypes instead of only 2 different phenotypes when using a binary alphabet. However, Goldberg (1990) has shown that schema processing is maximum with binary alphabets. Therefore, encoding integers using binary alphabets can be advantageous if we ensure that the phenotypes are encoded in a natural way (Goldberg, 1991). Thus, in this work we focus on representing integers using binary representations.

2.3.1 Binary Encoding

When using the *binary encoding*, each integer value $x_p \in \Phi_p = \{1, 2, \ldots, x_{p,max}\}$ is represented by a binary string $\boldsymbol{x}_g$ of length $l = \log_2(x_{p,max})$. The genotype-phenotype mapping f_g is defined as

$$x_p = f_g(\boldsymbol{x}_g) = \sum_{i=0}^{l-1} 2^i x_{g,i} \ ,$$

with $x_{g,i}$ denoting the ith bit of $\boldsymbol{x}_g$.

When using the binary encoding the contribution of the alleles to the phenotype is exponentially scaled. Therefore, GEAs solve the alleles serially starting from the most salient alleles. This effect can be modeled using the *domino convergence* model from Rudnick (1992). A result of the domino convergence is that low salient alleles start to drift and that they are fixed randomly before they can be reached by the solving process (Goldberg & Segrest, 1987; Asoh & Mühlenbein, 1994; Thierens, Goldberg, & Pereira, 1998).

Furthermore, the encoding has problems associated with the *Hamming cliff* (Schaffer, Caruana, Eshelman, & Das, 1989). The Hamming cliff describes the effect that some neighboring phenotypes (the phenotypes have a distance of one) are represented by completely different genotypes (the distance between the genotypes is much larger than one). The distance d between two genotypes $\boldsymbol{x}_g$ and $\boldsymbol{y}_g$ is defined by using the Hamming distance as $d_{\boldsymbol{x}_g,\boldsymbol{y}_g} = \sum_{i=0}^{l-1} |x_{g,i} - y_{g,i}|$ and denotes the number of different alleles in the two genotypes. The distance between two phenotypes x_p and y_p is defined as $d_{x_p y_p} = |x_p - y_p|$. An example for the use of the binary encoding is given in Table 1.

2.3.2 Gray Encoding

To overcome problems with the Hamming cliff and the different contribution of the alleles when using the binary encoding, the *gray encoding* was developed (Caruana & Schaffer, 1988; Schaffer, Caruana, Eshelman, & Das, 1989). When using gray encoding the average contribution of an allele to the represented integer is the same for all alleles in the bitstring.

The gray encoded bitstring itself can be constructed in two steps. At first, the phenotype is encoded using the binary encoding, and subsequently the binary encoded string can be converted into the corresponding gray encoded string. The binary string $\boldsymbol{x} \in \{0,1\}^l = \{x_1, x_2, \ldots, x_l\}$ is converted to the corresponding gray code $\boldsymbol{y} \in \{0,1\}^l = \{y_1, y_2, \ldots, y_l\}$ by the mapping $\gamma : \mathbb{B}^l \to \mathbb{B}^l$:

$$y_i = \begin{cases} x_i & \text{if } i = 1, \\ x_{i-1} \oplus x_i & \text{otherwise,} \end{cases}$$

where $\oplus$ denotes addition modulo 2. The decoding of a gray encoded string is as follows:

$$x_i = \bigoplus_{j=1}^{i} y_j,$$

for $i = \{1, \ldots l\}$. A gray encoded string has the same length l as a binary encoded string and the encoding is redundancy-free. Furthermore, the representation overcomes the problems with the Hamming cliff. Every two neighboring phenotypes ($d_{\boldsymbol{x}_p,\boldsymbol{y}_p} = 1$) are encoded by neighboring genotypes ($d_{x_g,y_g} = 1$). This property gives gray encoding an advantage over the binary encoding when using mutation-based search operators (compare also subsection 4.1). Table 1 illustrates an example for the use of the gray encoding.

Table 1: An example for using binary and gray encodings

x_p	0	1	2	3	4	5	6	7
binary	000	001	010	011	100	101	110	111
gray	000	001	011	010	110	111	101	100

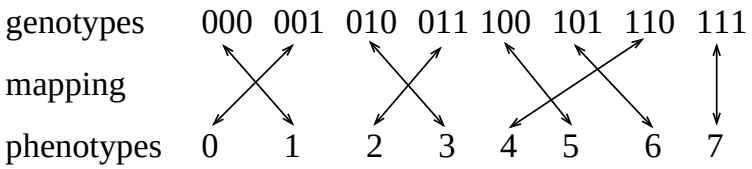


Figure 2: A random genotype-phenotype mapping

3 Performance of Crossover-based GAs using Binary Representations

In the following we show for the integer one-max problem how the performance of selectorecombinative GAs depends on the used representation.

3.1 Counting the Number of Binary Representations of Integers

We want to calculate the number of different genotype-phenotype mappings f_g.

If we use a redundancy-free encoding the number of genotypes is the same as the number of phenotypes. When using a binary representation of length l we are able to represent 2^l different phenotypes using a bitstring of length l. Therefore, the number of possible genotype-phenotype mappings is $2^l!$. The number of different representations increases exponentially with increasing l. One example for a possible genotype-phenotype mapping is given in Figure 2.

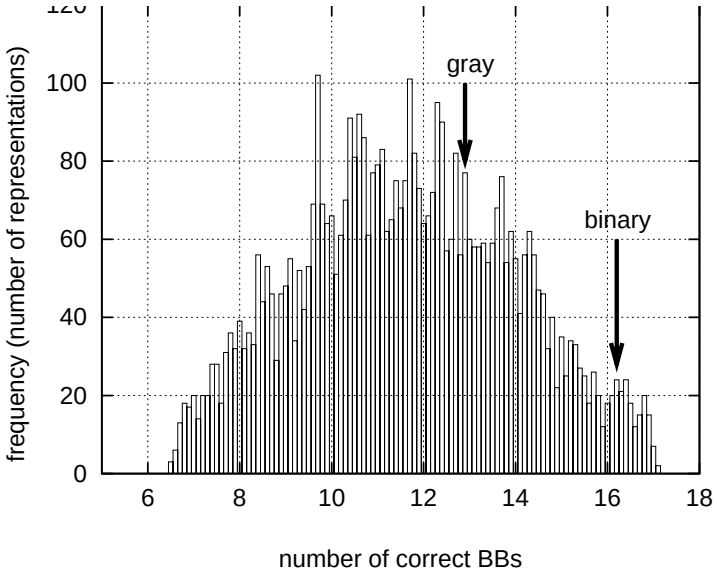
If we use a binary representation and we encode eight different phenotypes with a genotype of length $l = 3$, there are $2^3! = 40\,320$ different representations. Encoding 16 different phenotypes with a bitstring of $l = 4$ already results in more than 10^{13} different genotype-phenotype mappings. Therefore, to be able to systematically investigate how GA's performance depends on the used encoding we must limit ourselves to a genotypic string length of $l = 3$ and assume without loss of generality that the phenotype $x_p = 0$ is always assigned to the individual $\boldsymbol{x}_g = 000$. Then, the number of different genotype-phenotype mappings is reduced to $(2^l - 1)! = 7! = 5040$. Every genotype-phenotype mapping represents a different representation.

3.2 Experimental Results

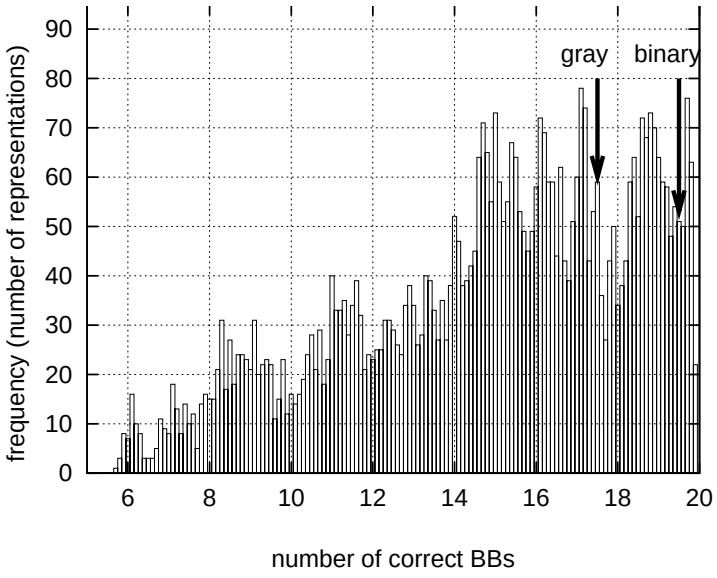
We present empirical results concerning the performance of selectorecombinative GAs using different types of representations for the integer one-max problem defined in subsection 2.2.

For our investigation we concatenate 20 integer one-max problems of size $l = 3$. Each of the 20 phenotypic integers $x_p \in \{0, \ldots, 7\}$ corresponds to 3 bits in the genotype. Therefore, the length of a genotype is $l_{\boldsymbol{x}_g} = 60$. The fitness of an individual is calculated as the sum over the fitness of the 20 BBs. The fitness of one BB is calculated according equation 1.

For our investigation we use a selectorecombinative GA using only uniform crossover and no mutation. For selection we use tournament selection without replacement of size 2. The population size is set either to $n = 20$ (Figure 3(a)) or $n = 40$ (Figure 3(b)). We performed 250 runs for each of the 5040 different genotype-phenotype mappings, and each run was stopped after the population



(a) population size $n = 20$



(b) population size $n = 40$

Figure 3: Experimental results of the frequency of the number of correct BBs at the end of a run for all possible genotype-phenotype mappings. We present results for 20 concatenated 3-bit problems. The genotype $\boldsymbol{x}_g = 000$ is always assigned to the phenotype $x_p = 0$ so there are $(2^3 - 1)! = 5040$ different possible genotype-phenotype mappings. We use a GA with tournament selection without replacement of size 2, uniform crossover, and no mutation. We perform 250 runs for each of the 5040 possible encodings.

was fully converged. A subproblem is correctly solved if the GA is able to find the best solution $x_p = 7$. The average number of correct BBs found at the end of the run gives us a measurement of the GA's performance using one specific representation. The more BBs which could be correctly identified at the end of the run, the higher the GA performance. As we limit ourselves to genotypes of length $l = 3$ and assign $x_p = 0$ always to $\boldsymbol{x}_g = 000$ there are 5040 different genotype-phenotype mappings (representations).

Figure 3 presents the results of our experiments for the integer one-max problem. We show the distribution of the number of correct BBs at the end of a GA run when using different types of genotype-phenotype mappings. The plots show results for all 5040 different genotype-phenotype mappings. The ordinate counts the number of genotype-phenotype mappings (representations) that allow a GA to correctly identify a certain number of BBs.

How can we interpret the data in Figure 3? Every bar indicates the number of different genotype-phenotype mappings that allow a GA to correctly identify a specific number of BBs. For example, the bar of height 77 at position 12.9 means that a GA correctly identifies on average between 12.85 and 12.95 BBs for 77 different genotype-phenotype mappings. The bar at position 17.0 means that there are only 7 (out of 5040) different genotype-phenotype mappings that allow a GA to correctly identify on average between 16.95 and 17.05 BBs. The plot shows that the differences in GA performance are large and that a GA with 20 individuals finds dependent on the used representation between 6.5 and 17.1 BBs out of 20 BBs.

If we compare Figure 3(a) with Figure 3(b) we see that with increasing population size there are still large differences in GA performance. We are able to correctly identify more BBs and there are many encodings that allow a GA with $n = 40$ to find on average more than 90% of all 20 BBs, but there are still some representations that result in very low GA performance.

We see that different representations, that means assigning the genotypes $\boldsymbol{x}_g \in \{0,1\}^3$ in a different way to the phenotypes $x_p \in \{0,\dots 7\}$, strongly change the performance of GAs. For some representations GA performance is high, whereas for some representations GA performance is low.

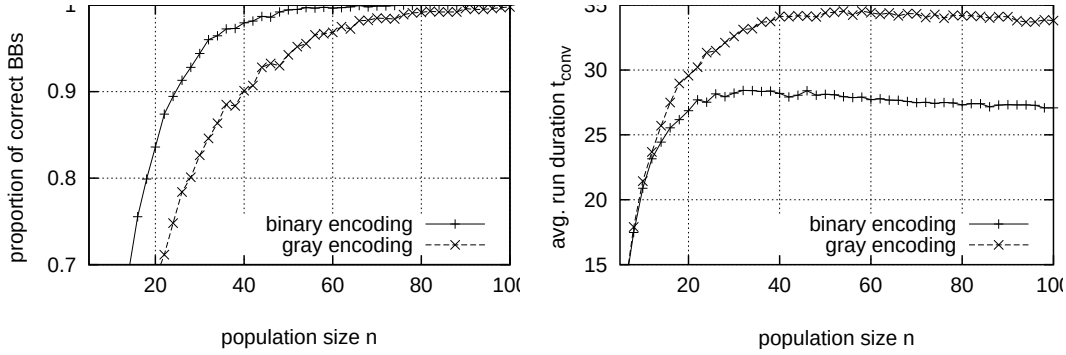
4 Performance of Binary and Gray Encoding

After we have seen that GA's performance strongly depends on the used representation, we focus in the following on the performance of two specific representations, namely gray and binary encoding.

When using binary representations of integers there are $2^l!$ different genotype-phenotype mappings. Gray and binary encoding are two specific representations. The arrows in Figure 3 indicate the performance of selectorecombinative GAs using these two types of encodings. A GA with $n = 20$ using gray encoding correctly identifies on average only 12.9 of the 20 BBs whereas when using binary encoding on average 16.2 out of the 20 BBs are found. It can be seen that GAs using binary encoding perform much better than GAs using gray encoding. With increasing population size n both encodings perform better but there is still a large performance difference between both encodings.

4.1 Experimental Results

To investigate the performance differences more closely we compare in Figure 4 GA performance when using binary encoding and gray encoding. We show the average proportion of correct BBs at the end of the run (Figure 4(a)) and the number of generations (Figure 4(b)) over the population size. As before, we concatenated $m = 20$ BBs of length $l = 3$. Furthermore, we use the same parameters for the GA as described in subsection 3.2 but only change the population size n. The



(a) Average proportion of correct BBs at the end of a GA run

(b) Number of generations

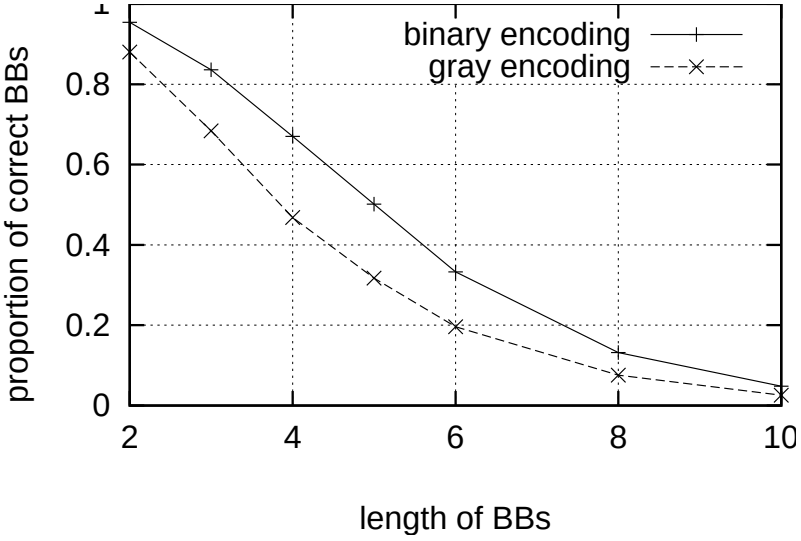
Figure 4: GA performance for the integer one-max problem. Each BB has length 3 and we concatenated $m = 20$ BBs. We show the average proportion of correct BBs at the end of a GA run (Figure 4(a)) and the average length of a run (Figure 4(b)). GAs using the binary encoding are able to find more BBs and converge after a shorter number of generations.

results confirm our previous observations and show that selectorecombinative GAs using the binary encoding not only find a higher proportion of BBs but also find the BBs in shorter time.

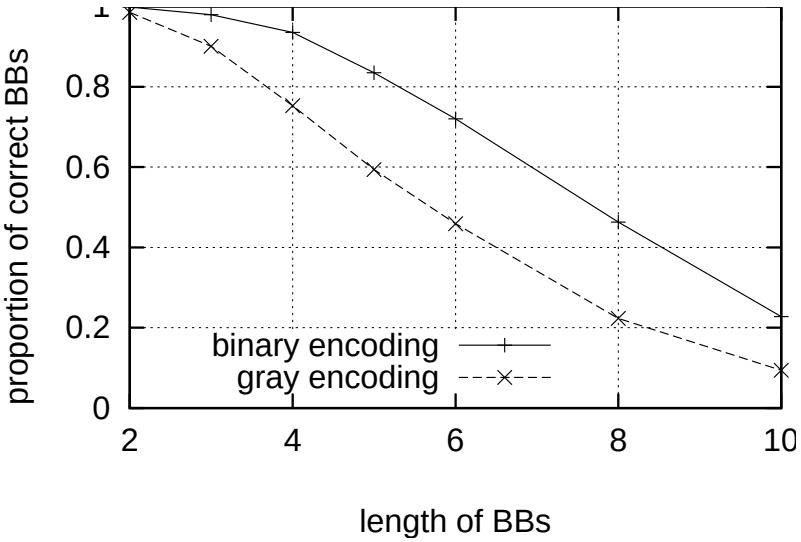
To investigate how GA performance varies over the length of the BBs we show in Figure 5 results for the proportion of correct BBs over the length l of the BBs. The length l of the BBs varies from $l = 2$ to $l = 10$. Therefore, the number of different integers that can be represented by a BB of length l varies from $2^2 = 4$ to $2^{10} = 1024$. As before we concatenate 20 subproblems of length l and use the same GA parameters as in subsection 3.2. The length of a genotype is $l_{\boldsymbol{x}_g} = 20 * l$.

GA performance declines with increasing length l of the BBs. For small problems ($l = 2$) GAs are able to correctly identify most of the 20 BBs, whereas for large problems ($l = 10$) only a small fraction of BBs can be found. Comparing gray and binary encoding shows that independently of the length l of the BBs GAs using binary encoding outperform GAs using gray encoding. With larger population size $n = 40$ (Figure 5(b)) GA performance increases. However, using binary encoding still results in better GA performance.

The performance differences between gray and binary encoding are surprising because we already noted in subsection 2.3 that the gray encoding has no problems with the Hamming cliff and the contribution of the alleles is uniform. Furthermore, other work has shown that the gray encoding shows some advantage in comparison to binary encoding. (Whitley, Rana, & Heckendorn, 1997; Rana & Whitley, 1997; Whitley & Rana, 1997; Rana & Whitley, 1998; Whitley, 1999; Whitley, 2000a; Whitley, 2000b). This work formulated a Free Lunch theorem for the use of gray encoding and mutation-based search approaches. GEAs using mutation as the main search operator perform better when using the gray encoding than when using the binary encoding. The proof actually shows that the number of local optima introduced by using the gray encoding is smaller than when using the binary encoding. However, this proof is not in contrast to the results obtained herein. When using the gray encoding all phenotypes with distance $d^p = 1$ are also neighboring genotypes ($d^g = 1$). Therefore, when using mutation-based search approaches and gray encoding, a small mutation of a genotype always results in the corresponding phenotype and the performance of mutation-based search approaches on easy problems must be higher when using gray encoding.



(a) population size $n = 20$



(b) population size $n = 40$

Figure 5: Proportion of correct BBs at the end of a GA run versus the length l of the BBs for the binary and gray encoding. The plots are for the integer one-max problem and a population size of $n = 20$ (Figure 5(a)) and $n = 40$ (Figure 5(b)). With increasing length l of the BBs the performance of GAs is reduced. When using gray encoding the performance of a GA declines much stronger than when using binary encoding.

However, in this work we focus not on mutation-based search methods but use crossover as the main search operator. Therefore, the proper method to measure how representations influence GA performance is to use schema analysis (Holland, 1975; Goldberg, 1989). The performance of selectorecombinative GAs is determined by the structure of the BBs and not the structure of the search space.

4.2 Schemata Analysis for Binary and Gray encoding

This subsection analyzes the fitness of the schemata resulting from the use of gray versus binary encoding. The analysis of the fitness of the schemata reveals that using binary encoding makes the integer one-max problem easier than using gray encoding. Therefore, the performance of selectorecombinative GAs is higher when using binary encoding.

We know that selectorecombinative GAs can solve a problem easily if the BBs are of low order and short defining length (Goldberg, 1989). A problem is completely easy if the size of the BBs $k = 1$. Then all schemata containing the global optimum are superior to their competitors. In contrast, a problem is difficult and deceptive of order k if all schemata of lower order than $k \in \{1, \dots l\}$ containing the global optimum are inferior to their competitors (Deb & Goldberg, 1994).

In Table 2, we present the average fitness of the schemata for the integer one-max problem using binary and gray encoding for $l = 3$. The numbers reveal that for the integer one-max problem with binary encoding all schemata containing the global optimum $\boldsymbol{x}_g = 111$ are superior to their competitors. Therefore, the integer one-max problem is very easy ($k = 1$) and selectorecombinative GAs show a high performance. The schema analysis for gray encoding reveals that the schemata containing the global optimum $\boldsymbol{x}_g = 100$ are not always superior to their competitors. Therefore, the problem is not completely easy any more, and GAs perform worse in comparison to using binary encoding.

The results show, that for easy problems like the integer one-max problem, the use of the binary encoding results in an easier problem in comparison to the use of the gray encoding. Therefore, selectorecombinative GAs perform better for easy problems when using the binary encoding.

5 Conclusions

This paper investigates how binary representations of integers influence the performance of selectorecombinative GAs.

It is well known, that when using representations every optimization problem can be separated into a genotype-phenotype mapping (the used representation) and a phenotype-fitness mapping (the optimization problem that should be solved). This paper illustrates for binary representations of integers, that the choice of a proper genotype-phenotype mapping is crucial for GA's success. The use of different representations can result in large differences in GA performance.

The binary and gray encoding are two well known possible binary representations of integers. Focusing on these two representations reveals for the easy integer one-max problem, that despite the fact that for the binary encoding neighboring genotypes do not correspond to neighboring phenotypes, selectorecombinative GAs perform better using binary encoding than using gray encoding. The performance difference is independent of the used population size and the length of the binary or gray encoded genotypic bitstring. An analysis of the fitness of the resulting schemata for the easy integer one-max problem explains the differences in performance. It reveals, that the use of the binary encoding results in BBs of lower order than the use of the gray encoding. When using

Table 2: Schemata fitness for the integer one-max problem using binary versus gray encoding. The integer one-max problem is completely easy for the binary encoding. Using gray encoding results in a more difficult problem, because some of the high quality schemata have the same fitness as misleading schemata.

	order	3	2			1			0
binary	schema	111	11*	1*1	*11	**1	*1*	1**	***
	fitness	**7**	**6.5**	**6**	**5**	**11**	**4.5**	**5.5**	**3.5**
	schema		01*	0*1	*01	**0	*0*	0**	
	fitness		2.5	2	3	3	2.5	1.5	
	schema		10*	1*0	*10				
	fitness		4.5	5	4				
	schema		00*	0*0	*00				
	fitness		0.5	1	2				
gray	schema	100	10*	1*0	*00	1**	*0*	**0	***
	fitness	**7**	**6.5**	**5.5**	**3.5**	**5.5**	**3.5**	**3.5**	**3.5**
	schema		11*	1*1	*11	0**	*1*	**1	
	fitness		4.5	**5.5**	**3.5**	1.5	**3.5**	**3.5**	
	schema		01*	0*1	*01				
	fitness		2.5	1.5	**3.5**				
	schema		00*	0*0	*00				
	fitness		0.5	1.5	**3.5**				

binary encoding the fitness of all schemata that contain the optimal solution is higher than the fitness of their strongest competitors, whereas when using gray encoding, some competing schemata have the same fitness than the schemata containing the optimal solution. Therefore, when using gray encoding the integer one-max problem becomes more difficult and the performance of selectorecombinative GAs is lower.

Our empirical analysis of GA performance has shown that the binary encoding results in higher performance than the gray encoding. However, Figure 3(a) reveals that there are representations which outperform the binary encoding. If we can theoretically describe the properties of these encodings and construct such representations, we would be able to increase the performance of GAs and solve integer problems more efficiently.

References

Asoh, H., & Mühlenbein, H. (1994). On the mean convergence time of evolutionary algorithms without selection and mutation. In Davidor, Y., Schwefel, H.-P., & Männer, R. (Eds.), *Parallel Problem Solving from Nature- PPSN III* (pp. 88–97). Berlin: Springer-Verlag.

Bagley, J. D. (1967). *The behavior of adaptive systems which employ genetic and correlation algorithms*. Doctoral dissertation, University of Michigan. (University Microfilms No. 68-7556).

Caruana, R. A., & Schaffer, J. D. (1988). Representation and hidden bias: Gray vs. binary coding for genetic algorithms. In Laird, L. (Ed.), *Proceedings of the Fifth International Workshop on Machine Learning* (pp. 153–161). San Mateo, CA: Morgan Kaufmann.

Deb, K., & Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, *10*, 385–408.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1990, September). *Real-coded genetic algorithms, virtual alphabets, and blocking* (IlliGAL Report No. 90001). Urbana, IL: University of Illinois at Urbana-Champaign.

Goldberg, D. E. (1991). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, *5*(2), 139–167. (Also IlliGAL Report 90001).

Goldberg, D. E., & Segrest, P. (1987). Finite Markov chain analysis of genetic algorithms. In Grefenstette, J. J. (Ed.), *Proceedings of the Second International Conference on Genetic Algorithms* (pp. 1–8). Hillsdale, NJ: Lawrence Erlbaum Associates.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Lewontin, R. C. (1974). *The genetic basis of evolutionary change*. Number 25 in Columbia biological series. New York: Columbia University Press.

Liepins, G. E., & Vose, M. D. (1990). Representational issues in genetic optimization. *Journal of Experimental and Theoretical Artificial Intelligence*, *2*, 101–115.

Mendel, G. (1866). Versuche über Pflanzen-Hybriden. In *Verhandlungen des naturforschenden Vereins*, Volume 4 (pp. 3–47). Brünn: Naturforschender Verein zu Brünn.

Rana, S., & Whitley, W. (1998). Search, binary representations, and counting optima. In *Proceeding of a workshop on Evolutionary Algorithms. Sponsored by the Institute for Mathematics and its Applications* (pp. 1–11). Colorado State University. in 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.

Rudnick, W. M. (1992). *Genetic algorithms and fitness variance with an application to the automated design of artificial neural networks*. Unpublished doctoral dissertation, Oregon Graduate Institute of Science & Technology, Beaverton, OR.

Schaffer, J. D., Caruana, R. A., Eshelman, L. J., & Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 51–60). San Mateo, CA: Morgan Kaufmann.

Thierens, D., Goldberg, D. E., & Pereira, Â. G. (1998). Domino convergence, drift, and the temporal-salience structure of problems. In of Electrical, I., & Engineers, E. (Eds.), *Proceedings of 1998 IEEE International Conference on Evolutionary Computation* (pp. 535–540). Piscataway, NJ: IEEE Service Center.

Vose, M. D. (1993). Modeling simple genetic algorithms. In Whitley, L. D. (Ed.), *Foundations of Genetic Algorithms 2* (pp. 63–73). San Mateo, CA: Morgan Kaufmann.

Whitley, D. (1999). A free lunch proof for gray versus binary encodings. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference: Volume 1* (pp. 726–733). San Francisco, CA: Morgan Kaufmann Publishers.

Whitley, D. (2000a). Functions as permutations: Implications for no free lunch, walsh analysis and statistics. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., & Schwefel, H.-P. (Eds.), *Parallel Problem Solving from Nature, PPSN VI* (pp. 169–178). Berlin: Springer-Verlag.

Whitley, D. (2000b). Local search and high precision gray codes: Convergence results and neighborhoods. In Martin, W., & Spears, W. (Eds.), *Foundations of Genetic Algorithms 6* (pp. unknown). San Francisco, California: Morgan Kaufmann Publishers, Inc. in press.

Whitley, D., & Rana, S. (1997). Representation, search, and genetic algorithms. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)* (pp. 497–502). AAAI Press/MIT Press.

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.