Binary Representations of Integers and the Performance of Selectorecombinative Genetic Algorithms

Franz Rothlauf*

Department of Information Systems
University of Bayreuth / Germany
franz.rothlauf@uni-bayreuth.de

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 illustrated that the used representation strongly influences the performance of GAs. The binary and Gray encoding are two examples for assigning bitstring genotypes to integer phenotypes. Focusing the 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 are proofs that show the superiority of Gray encoding. However, the performance of selectorecombinative GAs using binary representations of integers is determined by the resulting building blocks and not by the structure of the search space resulting from the Hamming distances between the individuals. Therefore, the performance difference between the encodings can be explained by analyzing the fitness of the resulting schemata.

1 Introduction

Integer optimization problems are important in many real-world applications. We know from previous work (Liepins & Vose, 1990; Rothlauf, 2001) 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

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

only use crossover and selection 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 for one specific problem. The paper ends with concluding remarks.

2 Binary Representations for Integer Optimization Problems

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 (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. 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.

2.2 An Integer Optimization Problem

This subsection defines the integer optimization problem we want to use for our investigations. We want to define an easy problem which is 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 our investigation we want to defined an integer-specific variation of the one-max problem as

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

2.3 Binary Representations of Integers

In this subsection we briefly discuss 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.

Binary Encoding When using the *binary encoding*, each integer value $x_p \in \Phi_p = \{0, 1, 2, \ldots, x_{p,max}\}$ 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 $\boldsymbol{x}_g$.

The binary 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 $d_{x_p y_p} = |x_p - y_p|$.

Gray Encoding To overcome problems with the Hamming cliff and the different contribution of the alleles to the fitness of an individual when using the binary encoding, the *Gray encoding* was developed (Gray, 1953; 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 $\boldsymbol{x} \in \{0,1\}^l = \{x_0, x_1, \ldots, x_{l-1}\}$ is converted to the corresponding Gray code $\boldsymbol{y} \in \{0,1\}^l = \{y_0, y_1, \ldots, y_{l-1}\}$ by the mapping $\gamma : \mathbb{B}^l \to \mathbb{B}^l$:

$$y_i = \begin{cases} x_i & \text{if } i = 0, \\ x_{i-1} \oplus x_i & \text{otherwise,} \end{cases}$$

where $\oplus$ denotes addition modulo 2. 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 ($d_{\boldsymbol{x}_g,\boldsymbol{y}_g} = 1$) are encoded by neighboring genotypes ($d_{x_p,y_p} = 1$). This property gives Gray encoding an advantage over the binary encoding when using mutation-based search operators (compare also subsection 4.1).

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.

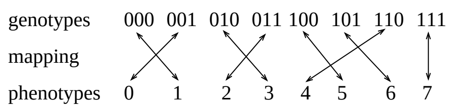


Fig. 1. A random genotype-phenotype mapping

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 is increasing exponentially with increasing l. One example for a possible genotype-phenotype mapping is given in Figure 1.

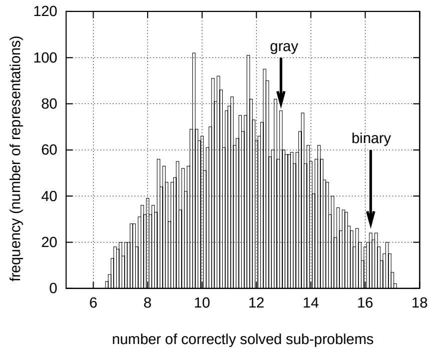
If we use a binary representation and we encode eight different phenotypes with a genotype of length $l = 3$, there are $2^3! = 40\,320$ different representations. Encoding 16 different phenotypes with a bitstring of $l = 4$ already results in more than 10^{13} different genotype-phenotype mappings. 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

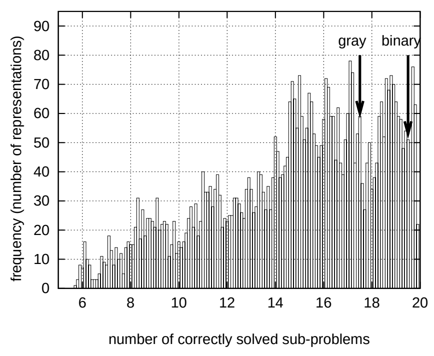
We present empirical results concerning the performance of selectorecombinative GAs using different types of representations for the integer one-max problem defined in subsection 2.2.

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 sub-problems. The fitness of one sub-problem is calculated according to 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 2(a)) or $n = 40$ (Figure 2(b)). We performed 250 runs for each of the 5040 different genotype-phenotype mappings, and each run was stopped after the population was fully converged. A sub-problem is correctly solved if the GA is able to find the best solution $x_p = 7$. The average number of correctly solved sub-problems at the end of the run gives us a measurement of the GA's performance using one specific representation. The more sub-problems can be solved correctly, 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).



(a) population size $n = 20$



(b) population size $n = 40$

Fig. 2. Experimental results of the frequency of the number of correctly solved sub-problems 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.

Figure 2 presents the results of our experiments for the integer one-max problem. We show the distribution of the number of correctly solved sub-problems 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 solve a certain number of sub-problems.

How can we interpret the data in Figure 2? Every bar indicates the number of different genotype-phenotype mappings that allow a GA to correctly solve a specific number of sub-problems. For example, the bar of height 77 at position

12.9 means that a GA correctly solves on average between 12.85 and 12.95 sub-problems 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 solve on average between 16.95 and 17.05 sub-problems. The plot shows that the differences in GA performance are large and that a GA with 20 individuals solves dependent on the used representation between 6.5 and 17.1 sub-problems out of 20.

If we compare Figure 2(a) with Figure 2(b) we see that with increasing population size there are still large differences in GA performance. The shapes of the distributions are similar and are shifted with increasing population size n towards a higher number of correctly solved sub-problems. To be able to illustrate how the performance of GAs depends on the different representations the population size n must be chosen relatively small. Using larger n would allow GAs to solve all 20 sub-problems (due to the easiness of the problem) and we would be not able to illustrate the performance differences using different types of representations.

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

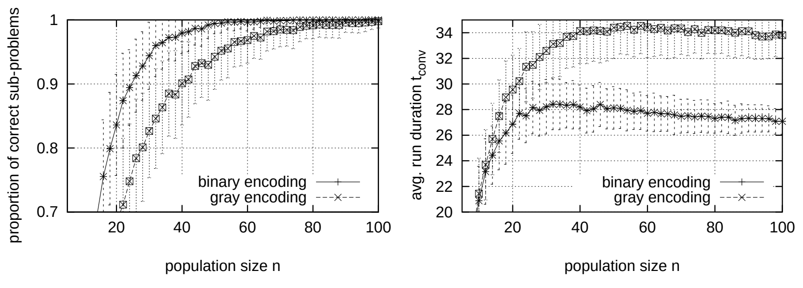
4 Performance of Binary and Gray Encoding

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 2 indicate the performance of selectorecombinative GAs using these two types of encodings. A GA with $n = 20$ using Gray encoding correctly solves on average only 12.9 of the 20 sub-problems whereas when using binary encoding on average 16.2 out of the 20 sub-problems are solved. 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

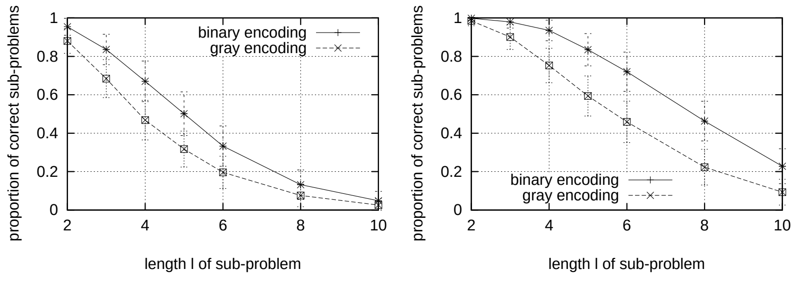
To investigate the performance differences more closely we compare in Figure 3 GA performance when using binary encoding and Gray encoding. We show the average proportion of correctly solved sub-problems at the end of the run (Figure 3(a)) and the number of generations (Figure 3(b)) over the population size. As before, we concatenated $m = 20$ sub-problems 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 results confirm our previous observations and show that selectorecombinative GAs using the binary encoding not only solve a higher proportion of correct sub-problems but also solve the sub-problems in shorter time.



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

(b) Number of generations

Fig. 3. GA performance for the integer one-max problem. Each sub-problem has length 3 and we concatenated $m = 20$ sub-problems. We show the average proportion of correctly solve sub-problems at the end of a GA run (Figure 3(a)) and the average length of a run (Figure 3(b)). GAs using the binary encoding are able to solve more sub-problems and converge after a shorter number of generations. The error bars indicate the standard deviation of the results.



(a) population size $n = 20$

(b) population size $n = 40$

Fig. 4. Proportion of correctly solved sub-problems at the end of a GA run versus the length l of the sub-problems for the binary and Gray encoding. The plots are for the integer one-max problem and a population size of $n = 20$ (Figure 4(a)) and $n = 40$ (Figure 4(b)). With increasing length l the performance of GAs is reduced. When using Gray encoding the performance of a GA declines much stronger than when using binary encoding. The error bars indicate the standard deviation of the results.

To investigate how GA performance varies over the length of the sub-problems we show in Figure 4 results for the proportion of correctly solved sub-problems over the length l. The length l of the sub-problems varies from $l = 2$ to $l = 10$. Therefore, the number of different integers that can be represented varies from $2^2 = 4$ to $2^{10} = 1024$. As before we concatenate 20 sub-problems 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 sub-problems. For small problems ($l = 2$) GAs are able to correctly solve most of the 20 sub-problems, whereas for large problems ($l = 10$) only a small fraction of sub-problems can be solved. Comparing Gray and binary encoding shows that independently of l GAs using binary encoding outperform GAs using Gray encoding. With larger population size $n = 40$ (Figure 4(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 uniformly. Furthermore, other work has shown that the Gray encoding shows some advantage in comparison to binary encoding. (Rana & Whitley, 1997; Whitley & Rana, 1997; Whitley, 1999). 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 herein which are obtained for selectorecombinative GAs and not for mutation-based search algorithms. 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.

However, in this work we focus not on mutation-based search methods but use crossover as main search operator. Therefore, the correct method to measure problem difficulty is to use schema analysis (Holland, 1975; Goldberg, 1989). The performance of selectorecombinative GAs is determined by the building blocks resulting from the used representation.

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. We perform a static schema analysis and do not consider the actual schemata a GA sees during the run. 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.

In Table 1, 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 easy 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 selectorecombinative GAs some easy problems like the presented integer one-max problem, are easier to solve when using the binary encoding as when using the Gray encoding. When using selectorecombinative

Table 1. 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**	**4**	**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**				

GAs, neither the Hamming distances between the individuals nor problems with Hamming cliffs are relevant for GA performance, but the schema analysis answers the question if a problem is easy or difficult.

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 results 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 for selectorecombinative GAs not Hamming distances between individuals, but schemata are important. We illustrate that the analysis of the fitness of the resulting schemata for the easy integer one-max problem can be used for explaining the differences in performance. It reveals, that the use of the binary encoding results in building blocks of lower order than the use of the Gray encoding. Therefore, when using Gray encoding the integer one-max problem is 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 2(a) reveals that there are representations that even outperform the binary encoding. If we can theoretically describe the properties of these encodings and systematically construct such representations, we would be able to increase the performance of GAs and solve integer problems more efficiently.

References

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.

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

Gray, F. (1953, March). Pulse code communications. U.S. Patent 2632058.

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

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

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.

Rothlauf, F. (2001). *Towards a theory of representations for genetic and evolutionary algorithms: Development of basic concepts and their application to binary and tree representations.* Doctoral dissertation, University of Bayreuth/Germany.

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.

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., & 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.