3.3 Locality

During the last decades, starting with work by Bagley (1967), Rosenberg (1967) and Cavicchio (1970), researchers recognized that the concept of building blocks is helpful for understanding the principles of selectorecombinative GAs, and is a key factor in determining successful use. A well designed GA should be able to preserve high quality building blocks, and increase their number over the generations (Goldberg 1989c).

When using the notion of building blocks in the context of representations, we must be aware that building blocks not only exist in the genotype, but also in the phenotype. A representation transforms the structure and complexity of the building blocks from the phenotype to the genotype, and therefore the structure and complexity of the building blocks can be different in the genotype and phenotype. This section provides the third, and final element, of a theory of representations for GEAs and investigates how the locality of a representation modifies the complexity of building blocks and changes the difficulty of an optimization problem. The locality of a representation describes how well genotypic neighbors correspond to phenotypic neighbors. The locality of a representation is high if genotypic neighbors correspond to phenotypic neighbors.

Previous work has indicated that high locality of a representation is necessary for efficient evolutionary search (Rothlauf and Goldberg 1999; Gottlieb and Raidl 1999; Gottlieb and Raidl 2000; Gottlieb et al. 2001; Rothlauf and Goldberg 2000). However, it remains unclear as to how the locality of a representation influences problem difficulty, and if high-locality representations always aid evolutionary search. The results show that high-locality representations do not modify the complexity of the problems they are used for. In contrast, when using low-locality representations the genotypic BBs can be different from the phenotypic BBs, and the complexity of the problem can be changed. Therefore, only high-locality representations can guarantee to reliably and predictably solve problems of bounded complexity.

We focus our investigation on how the locality of a representation influences problem difficulty and do not develop models for problem difficulty but use selected models presented in Sect. 2.3. In the context of evolution strategies (Bäck and Schwefel 1995), previous work developed the concept of causality (Igel 1998; Sendhoff et al. 1997b; Sendhoff et al. 1997a) as a measurement of problem difficulty. Basically, both concepts, causality and locality, address similar aspects as they describe how well the distances between individuals are preserved when mapping the phenotypes on the genotypes. However, in this study we do not classify the difficulty of problems but only describe how the difficulty of a problem is changed by the encoding.

In the following subsection, the influence of representations on problem difficulty is discussed. We illustrate why it is helpful to use encodings that do not modify problem difficulty for solving problems of bounded difficulty. Section 3.3.2 introduces the concept of locality and shows how locality depends on the metrics that are defined for the genotypes and phenotypes. In Sects. 3.3.3 and 3.3.4, we review the concept of problem difficulty introduced by Jones and Forrest (1995), and show how the locality of a representation influences problem difficulty. Sect. 3.3.5 introduces the distance distortion of a representation and illustrates that high locality is necessary for an encoding to have low distance distortion. In Sect. 3.3.6, we show for the fully easy one-max problem that low-locality representations make the problem more difficult. The results of the theoretical investigation are verified and illustrated in Sect. 3.3.7 by an empirical investigation for the one-max and deceptive trap problem. The results show that for high-locality representations the one-max problem remains fully easy, and the fully deceptive trap remains fully difficult. In contrast, when using low-locality representations the fully easy one-max problem becomes more difficult to solve for GEAs, whereas the deceptive trap becomes easier. The section ends with concluding remarks.

3.3.1 Influence of Representations on Problem Difficulty

We discuss the influence of representations on problem difficulty and why representations should preserve the complexity of a problem when assigning the genotypes to the phenotypes.

It was already recognized by Liepins and Vose (1990) that by using different representations the complexity of a problem can be completely changed. Changing the complexity of a problem means that the difficulty of the problem and therefore the structure of the BBs is changed (see Sect. 2.3). However, representations that modify BB-complexity are problematic as they do not allow us to reliably solve problems of bounded difficulty. If the complexity of BBs is changed by the encoding, some problems become easier to solve, whereas others become more difficult. To predict which problems become easier and which do not is only possible if detailed knowledge about the optimization problem exists (compare also Sects. 3.3.4 and 4.4.3). To ensure that problems of bounded complexity, that means easy problems, can be reliably solved by GEAs, representations that preserve the complexity of the building blocks should be used.

Liepins and Vose (1990) showed that for a fully deceptive problem $f(\boldsymbol{x}) = f_p(f_g(\boldsymbol{x}))$ there is a transformation T such that the function $g(\boldsymbol{x}) = f[T(\boldsymbol{x})]$ becomes fully easy. This means that every fully deceptive problem could be transformed into a fully easy problem by introducing a linear transformation T. In general, the difficulty of a problem could easily be modified by using different linear transformations T. Liepins and Vose concluded that their results underscore the importance of selecting good representations, and that good representations are problem-specific. Therefore, in order to find the best representation for a problem, it is necessary to know how to optimize the problem and what the optimal solution for the problem is.

To find the best representation for a specific problem, Liepins and Vose proposed using adaptive representations that “simultanously search for representations at a metalevel" (Liepins and Vose 1990, p. 110). These representations should autonomously adapt to the problem and encode it in a proper way (compare Rothlauf et al. (2000)). Initial steps in this direction were made by Goldberg et al. (1989) with the development of the messy GA. These kinds of GAs use an adaptive encoding that adapts the structure of the representation to the properties of the problem. This approach, however, burdens the GA not only with the search for promising solutions, but also the search for a good representation.

Therefore, we go back one step and ask the question, what kind of encoding should be used if there is no a priori knowledge about the problem, and no adaptive encoding should be used. Users who simply want to solve their problems by using a GEA are confronted with this kind of problem. They have no knowledge about the problem, and they do not want to do experiments to find out what structure the problem has, what the promising areas in the search space are, and what kind of representations make the problem most easy. They just want to be sure that the GEA can solve their problem as long as the complexity of their problem is bounded and the used GEA is able to solve it. One solution for their problems are representations that preserve BB-complexity. Using these types of representations means that the problem has the same difficulty in the genotypic as in the phenotypic space. Then users can be sure that the representation does not increase the complexity of the problem, and that the GEA reliably solves their problem.

Wanting representations to preserve BB-complexity raises the question of, why are we especially interested in encodings that preserve complexity? Is it not more desirable to construct encodings that reduce problem complexity, as Liepins and Vose propose? The answer is yes and no. Of course, we are interested in encodings that reduce problem difficulty. However, in general it is not possible to construct a representation that reduces the complexity for all possible problem instances. One result of the no-free-lunch theorem (Wolpert and Macready 1995) is that if some problem instances become easier by the use of a representation, there are other types of problem instances that necessarily become more difficult.

Therefore, we want to at least ensure that the representation does not make problems of bounded difficulty – these are the class of problems we are interested in – more difficult to solve. However, as shown in Sects. 3.3.6 and 3.3.7, encodings that do not preserve problem complexity always make fully easy problems more difficult. Therefore, a phenotypically easy problem could even become so difficult by using a "bad" encoding that it can not be solved efficiently any more. Of course, representations that do not preserve problem complexity make fully difficult problems more easy, but as we are interested in solving only problems of bounded difficulty, and not all types of problems, this is not important to us. For difficult problems, like the needle in the haystack problem, or the fully deceptive trap problem, the complexity of the problem is not bounded, and therefore, we are in general not interested in solving these kinds of problems with GEAs.

The solution for all our problems would be an "optimal" representation that preserves problem complexity for problems of bounded difficulty *and* reduces the complexity for all other problems. But reaching this aim is far beyond the scope of this work. Furthermore, we believe that the no no-free-lunch theorem does not allow us to get such a free lunch for every problem.

Finally, we want to come back to the results of Liepins and Vose (1990) and illustrate the problems with representations that change the difficulty of problems. The transformation T, which can be interpreted as a genotype-phenotype mapping, can modify the complexity of a problem in such a way that a fully difficult deceptive trap problem becomes a fully easy one-max problem. But, using the same transformation T for a fully easy one-max problem can result in a fully deceptive trap (compare Fig. 4.1(a)). Therefore, by using this representation, we are able to solve a deceptive trap, but not the one-max problem any more. If we want to solve both types of problems we must know a priori what the problem is and adjust the representation according to the problem. However, if we do not know the problem a priori, and if we want to make sure that we can solve at least problems of bounded difficulty reliably, we must use representations that do not modify problem difficulty.

In the following subsection, we define the locality of a representation formally and show in Sect. 3.3.4 that low locality is necessary for a representation to not modify problem difficulty.

3.3.2 Metrics, Locality, and Mutation Operators

In Sects. 2.1.1 and 2.1.2, we introduced the concept of a representation which assigns genotypes $x^g \in \Phi_g$ to corresponding phenotypes $x^p \in \Phi_p$. In the following paragraphs, we introduce the concept of locality and describe how the locality of a representation is based on the metric used for Φ_g and Φ_p.

When using search algorithms, a metric has to be defined on the search space Φ. Based on the metric, the distance d_{x_a,x_b} between two individuals $x_a \in \Phi$ and $x_b \in \Phi$ describes how similar the two individuals are. The larger the distance, the more different two individuals are. In general, different metrics can be defined for the same search space. Different metrics result in different distances and different measurements of the similarity of solutions.

Two individuals are neighbors if the distance between two individuals is minimal. For example, when using the Hamming metric (Hamming 1980) for binary strings the minimal distance between two different individuals is $d_{min} = 1$. Therefore, two individuals x_a and x_b are neighbors if the distance $d_{x_a,x_b} = 1$.

If we use a representation f_g there are two different search spaces, Φ_g and Φ_p. Therefore, different metrics can be used for the phenotypic and the genotypic search space. In general, the metric used on the phenotypic search space Φ_p is determined by the specific problem that should be solved and describes which problem solutions are similar to each other. In contrast, the metric defined on Φ_g is not given a priori but depends on the used genotypes. As different genotypes can be used to represent the same phenotypes, different metrics can be defined on Φ_g. Therefore, in general, different metrics are used for Φ_p and Φ_g which imply a different neighborhood structure in Φ_g and Φ_p. For example, when encoding integers using binary strings the phenotype $x^p = 5$ has two neighbors, $y^p = 6$ and $z^p = 4$. When using the Hamming metric, the corresponding binary string $x^g = 101$ has three different neighbors, $a^g = 001$, $b^g = 111$, and $z^g = 100$ (Caruana and Schaffer 1988).

The locality of a representation describes how well neighboring genotypes correspond to neighboring phenotypes. The locality of a representation is high if all neighboring genotypes correspond to neighboring phenotypes. In contrast, the locality of a representation is low if some neighboring genotypes do not correspond to neighboring phenotypes. Therefore, the locality d_m of a representation can be defined as

$$d_m = \sum_{d^g_{x,y}=d^g_{min}} |d^p_{x,y} - d^p_{min}|, \tag{3.23}$$

where $d^p_{x,y}$ is the phenotypic distance between the phenotypes x^p and y^p, $d^g_{x,y}$ is the genotypic distance between the corresponding genotypes, and d^p_{min}, resp. d^g_{min} is the minimum distance between two (neighboring) phenotypes, resp. genotypes. Without loss of generality we want to assume that $d^g_{min} = d^p_{min}$. For $d_m = 0$ all genotypic neighbors correspond to phenotypic neighbors and the encoding has perfect (high) locality.

We want to emphasize that the locality of a representation does not only depend on the representation f_g, but also on the metric that is defined on Φ_g and the metric that is defined on Φ_p. f_g only determines which phenotypes are represented by which genotypes and says nothing about the similarity between solutions. Before we are able to describe the locality of a representation a metric must be defined on Φ_g and Φ_p.

In the remaining paragraphs, we briefly discuss how the mutation operator used for genetic search determines the metric and the distances that are used for the genotypic space Φ_g. Based on the metric defined on the genotypic search space Φ_g, search operators like mutation or crossover can be defined. In EAs, and most of the individual-based search heuristics, like simulated annealing, tabu search, and others, the search operator mutation is designed to create new solutions (offspring) with similar properties as its/their parent(s) (Doran and Michie 1966). In most local search methods, mutation creates offspring that have a small or sometimes even minimal distance to their parents (for example the bit-flipping operator for binary representations). Therefore, mutation operators and metrics can not be developed independently of each other but determine each other. A metric defines the mutation operator and the used mutation operator determines the metric. As the search operators are applied to the genotypes, the metric that is used on Φ_g is relevant for the definition of mutation operators.

The basic idea behind using mutation-based search approaches is that the structure of the fitness landscape should guide the search heuristic to the high-quality solutions (Manderick et al. 1991), and that the optimal solution can be found by performing small iterated changes. It is assumed that the high-quality solutions are not isolated in the search space but grouped together (Christensen and Oppacher 2001; Whitley 2002). Therefore, better solutions can easily be found by searching around already found good solutions. The search steps must be small because too large search steps would result in a randomization of the search, and guided search around good solutions would become impossible. In contrast, when using search operators that perform large steps in the search space it would not be possible to find better solutions by searching around already found good solutions but the search algorithm would jump randomly around the search space (compare also Sect. 3.1.2).

3.3.3 Phenotype-Fitness Mappings and Problem Difficulty

As described in Sect. 2.1.2 the difficulty of a problem depends on the phenotype-fitness mapping f_p. Furthermore (compare Sect. 2.3), the difficulty of a problem depends on the used search method and the metric that is defined on the phenotypic search space Φ_p. The metric defined on Φ_p determines which individuals are similar to each other and depends on the used main search operator. Both determinants of problem difficulty, the phenotype-fitness mapping f_p and the metric defined on Φ_p, are given a priori by the character of the optimization problem that should be solved and by the used search method.

In Sect. 3.3.2, we described that the mutation operator and the used metric determine each other. Different mutation operators imply different metrics. As problem difficulty depends not only on f_p but also on the metric defined on Φ_p the difficulty of a problem is not absolute but depends on the used metric respectively search operator. The use of different metrics and search operators result in a different problem difficulty. Consequently, the difficulty of a problem can only be defined with respect to a search operator. It makes no sense to say a problem is either easy or difficult if the used search operator is not taken into account.

In Sect. 2.3.2, we gave a short review of some measurements for problem difficulty in the context of GEAs. In the following subsections, we want to use the classification of problem difficulty from Jones and Forrest (1995), which is based on the correlation analysis, for describing how the locality of a representation influences GEA performance. The difficulty of an optimization problem is determined by how the fitness values are assigned to the phenotypes and what metric is defined on the phenotypes. Combining both aspects we can measure problem difficulty by the fitness-distance correlation coefficient $\rho_{FDC} \in [-1, 1]$ of a problem (Jones 1995; Jones and Forrest 1995). ρ_{FDC} measures the correlation between the fitnesses of search points and their distances to the global optimum. We want to distinguish between three different classes of problem difficulty:

1. The fitness difference to the optimal solution is positively correlated with the distance to the optimal solution. With lower distance the fitness difference to the optimal solution decreases. As the structure of the search space guides local search methods to the optimal solution such problems are easy for mutation-based search.
2. There is no correlation between the fitness difference and the distance to the optimal solution. The fitness values of neighboring individuals are uncorrelated and the structure of the search space provides no information about which solutions should be sampled next by the search method.
3. The fitness difference is negatively correlated to the distance to the optimal solution. Therefore, the structure of the search space misleads a local search method to sub-optimal solutions.

The three different classes of problem difficulty are illustrated in Fig. 3.13. We show how the fitness difference $|f_{opt} - f|$ depends on the distance d to the optimal solution. In the following paragraphs, we want to discuss these three classes in some more detail.

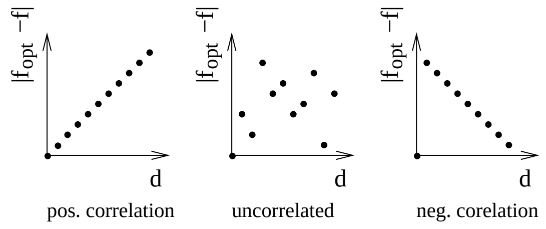

Figure 3.13. Different classes of problem difficulty

Problems are easy for mutation-based search if there is a positive correlation between an individuals' distance to the optimal solution and the difference between its fitness and the fitness of the optimal solution. Many test problems that are commonly used for EAs like the sphere and corridor models for evolution strategies or the one-max problem for genetic algorithms show this behavior. Such problems are easy for local and crossover-based search methods as the search is guided to the optimal solution by the structure of the fitness landscape.

Problems become much more difficult if there is no correlation between the fitness difference and the distance to the optimal solution. Then, the fitness landscape does not guide a mutation-based search method to the optimal solution. No search heuristics can use information about a problem which was collected in prior search steps to determine the next search step. Therefore, all reasonable search algorithms show the same performance as no useful information (information that indicates where the optimal solution can be found) is available in the problem. Because all search strategies are equivalent, also random search is an an appropriate search method for such problems. Random search uses no information and performs well on these types of problems.

Problem difficulty is maximal for mutation-based search methods if the fitness landscape leads the search method away from the optimal solution. Then, the distance to the optimal solution is negatively correlated to the fitness difference between an individual and the optimal solution. Because mutation-based search finds the optimal solution by performing iterated small steps in the direction of better solutions, all mutation-based search approaches must fail as they are mislead. All other search methods that use information about the fitness landscape also fail. The most effective search methods for such problems are those that do not use information about the structure of the search space but search randomly like random search. The most prominent example for such types of problems are deceptive traps. Such problems are commonly used to perform a worst-case analysis for EAs.

Although we use this problem classification for investigating the influence of locality on problem difficulty, we want to emphasize that in general this problem classification is not relevant for most of the real-world problem instances. Only problems of class one can be solved efficiently using EAs or local search as this problem class guides the local search methods (like mutation-based EAs) to the good solutions. In contrast, for problems of class two, mutation-based search methods perform the same as random search, and for problems of class three random search performs even better. This situation is not in contrast to the observed good performance of EAs on many real-world problem instances. EAs show a good performance as most of the real-world problems are easy problems and belong to class one (compare also Sect. 3.3.1). In general, for real-world problems the fitness values of neighboring solutions are correlated, and high-quality and low-quality solutions are grouped together. Therefore, fitness landscapes that are uncorrelated, or even deceptive, are uncommon in real world.

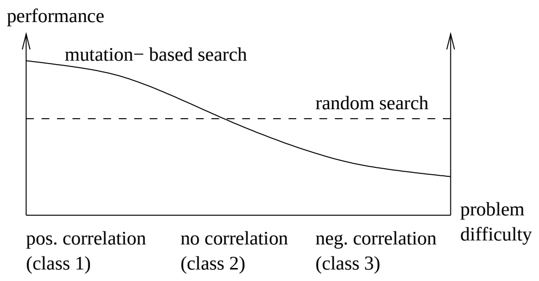

Figure 3.14. Performance of mutation-based EA search versus random search

This situation is illustrated in Fig. 3.14. We know from the no-free-lunch theorem that all search methods show on average the same performance over all possible problem instances (Wolpert and Macready 1995; Whitley 2000a).

Furthermore, we know that the performance of random search remains constant over all problem instances and that mutation-based evolutionary search performs well on problems of class one. Consequently, it must show low performance on other problem instances (class 3). As the performance of mutation-based search is "biased" towards problems of class one, many real-world instances can efficiently be solved using mutation-based EAs.

3.3.4 Influence of Locality on Problem Difficulty

In Sect. 3.3.1, we described how representations can change the character and difficulty of optimization problems. In the following paragraphs, we discuss high versus low-locality representations and examine how the locality of a representation influences problem difficulty.

Low versus High-Locality Representations

We have seen in Sect. 3.3.2 that the metric defined on Φ_p can be different from the metric defined on Φ_g. As the locality of a representation measures how well the phenotypic metric corresponds to the genotypic metric, it is possible to distinguish between high and low-locality representations. Representations have high locality if neighboring genotypes correspond to neighboring phenotypes. In contrast, representations have low locality if neighboring genotypes do not correspond to neighboring phenotypes. Figure 3.15 illustrates the difference between high and low-locality representations. In this example, we assume that there are 12 different phenotypes (a-l) and that there is a metric defined on Φ_p (in this example the Euclidean distance). Each phenotype (lower case symbol) corresponds to one genotype (upper case symbol). The representation f_g has perfect (high) locality if neighboring phenotypes correspond to neighboring genotypes. Then a mutation step has the same effect in the phenotypic and genotypic search space.

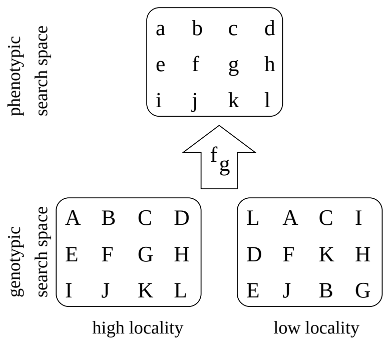

Figure 3.15. High versus low-locality representations

As we assume in our considerations that f_g is a one-to-one mapping every phenotype is represented by exactly one genotype and there are $|\Phi_g|! = |\Phi_p|!$

different representations. $|\Phi_g|$ is the size of the genotypic search space. Each of these many different representations assigns the genotypes to the phenotypes in a different way. A common example are different representations for representing integer phenotypes using binary strings. Both, binary and Gray encoding, represent integers using binary strings of the same length but they differ in which phenotype is represented by which genotype.

Investigating the relationship between different representations (how the genotypes are assigned to the phenotypes) and the used mutation operator (which is based on the genotypic metric) reveals that a different assignment of genotypes to phenotypes can be equivalent to the use of a different metric for Φ_g. This effect is known as the isomorphism of fitness landscapes (Reeves 1999). For example, it can be shown that the use of a simple bit-flipping operator (which induces the Hamming metric) for Gray encoded problems is equivalent to the use of the complementary crossover operator (which induces a different "non-Hamming" metric) for binary encoded problems (Reeves 2000). Both metric-representation combinations result in the same fitness landscape and therefore in the same performance of mutation-based search.

Influence on Problem Difficulty

In the following paragraphs, we examine how the locality of a representations influences the performance of GEAs. A representation transforms the phenotypic problem f_p with a given phenotypic problem difficulty into a genotypic problem $f = f_p \circ f_g$ with a resulting genotypic problem difficulty that can be different from the phenotypic problem difficulty. We use the problem classification described in Sect. 3.3.3.

We have seen that the phenotypic difficulty of an optimization problem depends on the metric that is defined on the phenotypes and the function f_p which assigns a fitness value to every phenotype. Based on the phenotypic metric a local or crossover-based search operator can be defined (for the phenotypes). By the use of a representation, which assigns a genotype to every phenotype, a new genotypic metric is introduced which can differ from the phenotypic metric. Therefore, the character of the search operator can also be different for genotypes and phenotypes. If the locality of a representation is high, then the search operator has the same effect on the phenotypes as on the genotypes. As a result, genotypic and phenotypic problem difficulty is the same and the difficulty of a problem remains unchanged by the use of an additional representation f_g. Easy phenotypic problems remain genotypically easy (compare the results presented in Fig. 3.21) and difficult phenotypic problems remain genotypically difficult (compare Fig. 3.22). Figure 3.16 (left) illustrates the effect of mutation for high-locality representations. The search operator (mutation) has the same effect on the phenotypes as on the genotypes.

The situation is different when focusing on low-locality representations. Here, the influence of the representation on the difficulty of a problem depends on the considered optimization problem. If the considered problem f_p

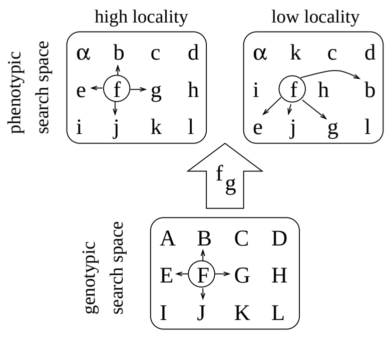

Figure 3.16. The effect of mutation for high versus low-locality representations

is easy (class 1) and the structure of the search space guides the search method to the optimal solution, a low-locality representation f_g randomizes the problem and makes the overall problem f more difficult. When using low-locality representations a small change in a genotype does not correspond to a small change in the phenotype, but larger changes in the phenotype are possible (compare Fig. 3.16 (right)). Therefore, when using low-locality representations, phenotypic easy problems of class one become on average genotypic problems of class two. Low-locality representations lead to a more uncorrelated fitness landscape and heuristics can no longer extract information about the structure of the problem. Guided search becomes more difficult as many genotypic search steps do not result in a similar individual but in a random one.

If the problem f_p is of class two, on average a low-locality representation does not change the problem class. Although the mutation-based search becomes more random search, the performance stays constant as random search and mutation-based search show the same performance for problems of class two. Of course, representations exist that can make a problem easier and result in an overall genotypic problem f of class one; however, there are only few of these and most of the low-locality representations simply modify the problem and do not create a fitness landscape of class one which leads the search method to the good solutions. On the other hand, there are also representations f_g that construct a problem f which misleads mutation-based search and transforms a problem of class two into class three. But as for low-locality representations that transform a problem from class two into class one, there are only few such representations.

Finally, we have to consider problems of class three. On average, the use of low-locality representations transforms such problems into problems of class two as the problems become more randomized. Then, mutation-based search is less misled by the fitness landscape and the problem difficulty for mutation-based search is reduced. On average, low-locality representations "destroy" the deceptiveness of class three problems and turn them into problems of class two.

Summarizing the results, we recognize that low-locality representations have the same effect as when using random search. Therefore, on average problems of class one become more difficult, and problems of class three more easy to solve. As most real-world problems belong to class one, the use of low-locality representations makes these problems more difficult. Therefore, we strongly encourage researchers to use high-locality representations for problems of practical relevance. Of course, low-locality representations make deceptive problems easier; however, these are problems which we do not expect to meet in reality and are only of theoretical interest.

3.3.5 Distance Distortion and Crossover Operators

We extend the notion of locality and introduce the *distance distortion* of an encoding. The concept of distance distortion is related to the concept of heritability which describes that a crossover operator should create new offspring that have similar properties to their parents. Appropriate measurements for heritability describe how well offspring take over advantageous features of their parents (Gottlieb and Raidl 1999).

When using recombination-based search, the locality concerning small changes d_m can be extended towards locality concerning small and large changes. The distance distortion d_c describes how well the phenotypic distance structure is preserved when mapping Φ_p on Φ_g:

$$d_c = \frac{2}{n_p(n_p-1)} \sum_{i=1}^{n_p} \sum_{j=i+1}^{n_p} |d^p_{x,y} - d^g_{x,y}|,$$

where n_p is the number of different individuals, $n_p = |\Phi_g| = |\Phi_p|$, and $d^g_{min} = d^p_{min}$. For $d_c = 0$ all phenotypic distances are preserved by the representation. We see that for $\Phi_g = \Phi_p$ high locality ($d_m = 0$) results in low distance distortion ($d_c = 0$). If, for example, our genotypic and phenotypic search space is binary, and the locality of the genotype-phenotype mapping is perfect, then all distances between the individuals are preserved. However, if we assume that $\Phi_g \neq \Phi_p$, then high locality is a necessary, but not sufficient condition for the genotype-phenotype mapping to have low distance distortion.

Figure 3.17 illustrates the difference between representations with high versus low distance distortion. The distance distortion d_c of a representation is low if the genotypic distances correspond to the phenotypic distances. If the distances between the genotypes and the corresponding phenotypes are different, then the distance distortion d_c of the representation is high.

It is of interest that the locality d_m and distance distortion d_c do not require the definition of genetic operators a priori. It is sufficient to define both based on the distance metrics used for Φ_g and Φ_p. The application of mutation to an individual should result in an offspring that is similar to its parent. Therefore, in many implementations, mutation creates offspring who have the lowest possible distance to the parent (for example the bit-flipping

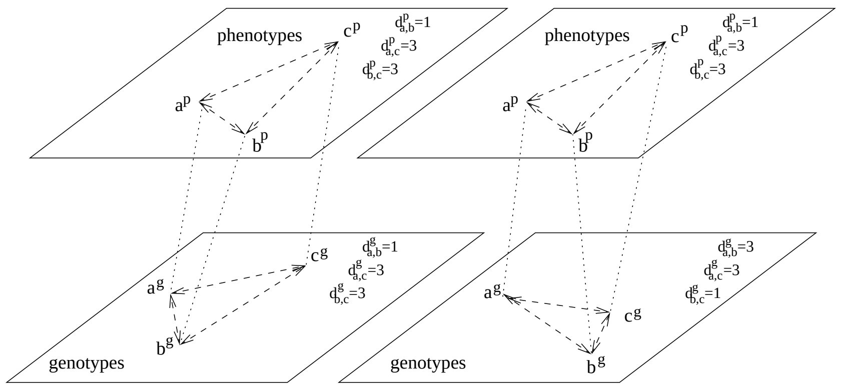

Figure 3.17. The figures illustrate the difference between representations with low versus high distance distortion. If the distance distortion $d_c = 0$ then the distances between the corresponding genotypes and phenotypes are the same. If $d_c > 0$ genotypic distances between individuals do not correspond to phenotypic distances between individuals.

operator for binary representations). High locality of a representation is a necessary condition for successful use of mutation-based search algorithms. Otherwise, low-locality encodings do not allow a guided search and GEAs using low-locality representations behave like random search.

The situation is similar when using crossover operators. The application of crossover operators should result in offspring where the distances between the offspring and its two parents are smaller than the distance between both parents. Common standard crossover operators, like n-point or uniform crossover show this behavior. The distances between genotypic offspring and parents are always lower, or equal to, the distances between both parents. However, if a representation has high distance distortion, the genotypic distances do not correspond to the phenotypic distances. Then, the phenotypic distances between offspring and parents are not necessarily smaller than the phenotypic distances between both parents. The application of crossover to genotypes does not result in offspring phenotypes that mainly consist of substructures of their parents' phenotypes. Therefore, the offspring is not similar to its parents and the use of crossover results in random search. We see that low distance distortion of a representation is a necessary condition for good performance of crossover-based GAs.

Examining the interdependencies between locality and distance distortion shows that high locality is a necessary condition for an encoding to have low distance distortion. When using standard crossover operators such as uniform or n-point crossover, the offspring could not inherit the properties of the parents if similar genotypes result in completely different phenotypes. If the encoding has low locality, the crossover operators would create offspring genotypes which are similar to the genotypes of the parents, but the resulting phenotypes would not be similar to the phenotypes of the parents. Thus, low locality of a representation would also result in high distance distortion.

In the following subsection, we show for the fully easy one-max problem that the problem only stays fully easy if a high-locality representation is used and $d_m = d_c = 0$. All other types of representations increase the difficulty of the problem.

3.3.6 Modifying BB-Complexity for the One-Max Problem

In this subsection, we investigate the influence of locality on GA performance. We show for the fully easy one-max problem (3.24) that the use of a high-locality representation does preserve problem difficulty whereas other types of representations reduce GA performance.

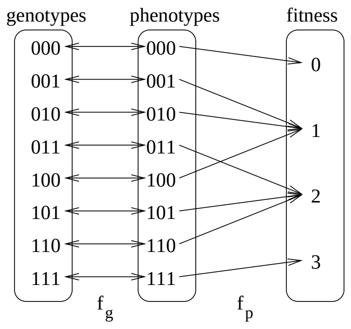

Figure 3.18. A representation for the bit-counting function with perfect locality and no distance distortion

For the one-max, or bit-counting problem, the function $f_p : \{0,1\}^l \rightarrow \mathbb{R}$ assigns to every individual $x^p \in \Phi_p$ the fitness value $\sum_{i=0}^{l-1} x_i^p$. As there are only l fitness values that are assigned to 2^l phenotypes the fitness function f_p is affected by redundancy (see Sect. 3.1.6). The genotype-phenotype mapping f_g is a non-redundant one-to-one mapping, and the genotypic space Φ_g, and the phenotypic space Φ_p, have the same size $|\Phi_g| = |\Phi_p| = 2^l$ and the same properties $\Phi_g = \Phi_p$. To simplify the investigation we want to assume, without loss of generality, that the phenotype with only ones is always represented by the genotype with only ones, and therefore is always the global optimum. In Fig. 3.18, a 3-bit one-max problem is illustrated. The encoding used, which can be described by the genotype-phenotype mapping f_g, has high locality ($d_m = 0$) and preserves the distances between the individuals when mapping the phenotypes to the genotypes ($d_c = 0$) as the genotype-phenotype mapping is the identity mapping $x^p = f_g(x^g) = x^g$. As a result, the phenotypic and genotypic problem complexity is the same.

We investigate how problem difficulty changes if we use a representation where $d_c \neq 0$. For measuring problem difficulty we use the fitness of the schemata (compare Sect. 2.3.2). The fitness of a schema h of size λ is defined as $f(h) = f(u, \lambda, l)$. It has length l, u ones, $\lambda - u$ zeros in the fixed positions, and $l - \lambda$ don't care positions. For the one-max problem the schema fitness in terms of the function values $f(u) = u$ can be calculated as follows:

$$f(u, \lambda, l) = \frac{1}{2^{l-\lambda}} \sum_{i=0}^{l-\lambda} \binom{l-\lambda}{i} (i + u).$$

For all schemata of size λ the difference between the fitness $f(\lambda, \lambda, l)$ of the best schemata with λ ones, and the fitness $f(\lambda - 1, \lambda, l)$ of its strongest competitor with $\lambda - 1$ ones is

$$d = \frac{1}{2^{l-\lambda}} \sum_{i=0}^{l-\lambda} \binom{l-\lambda}{i} > 0.$$

Thus, all schemata h that contain the global optimum x^{opt} (a string of only ones) are superior to their competitors, and the one-max problem is phenotypically fully easy.

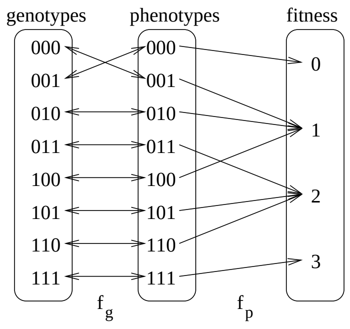

Figure 3.19. A low-locality representation for the one-max problem which does not preserve the distances between the individuals ($d_c \neq 0$), and therefore modifies BB-complexity. The distance distortion $d_c = \frac{2}{7*8} * 12 = 3/7 \neq 0$.

We investigate how the problem complexity changes if the distances between the individuals are changed by using a low-locality representation. For example, the genotype-phenotype mapping of two genotypes x^g and y^g which correspond to the phenotypes x^p and y^p, is changed such that afterwords $x^g = 000$ represents $y^p = 001$, and $y^g = 001$ represents $x^p = 000$ (see Fig. 3.19). Beginning with the high-locality encoding illustrated in Fig. 3.18, there are three different possibilities when changing the mapping of two individuals:

- Both individuals x^g and y^g have the same number of ones ($u_{x^g} = u_{y^g}$)
- Both individuals have a different number of ones, and the number of different positions d_{x^g,y^g} in the two individuals x^g and y^g is the same as the number of different ones ($d_{x^g,y^g} = u_{x^g} - u_{y^g}$)
- Both individuals have a different number of ones, and the number of different positions is higher than the number of different ones ($d_{x^g,y^g} > u_{x^g} - u_{y^g}$)

In the following paragraphs, we investigate how the difficulty (measured using the notion of schemata) of the problem is changed for these three situations.

If f_g is modified for two genotypes x^g and y^g that have the same number of ones in the string, then the corresponding fitness values remain unchanged ($f(x^g) = f(y^g)$). Therefore, the fitness of the schemata, and the difficulty of the problem both remain constant. For example, we can change the mapping of the genotypes 1001 and 0101 for a 4 bit one-max problem. Both individuals have two ones in the string and their fitness is two. The difficulty of the problem remains unchanged.

f_g could be modified for two individuals x^g and y^g that have a different number of ones, and therefore different fitness values. We assume that the number of different positions in the two individuals is the same as the number of different ones ($d_{x^g,y^g} = u_{x^g} - u_{y^g}$). Before the change, the individual x^g has l ones and therefore fitness l; y^g has h ones and fitness h. We want to assume $h > l$. After the change, x^g has fitness h although it has only l ones, whereas y_g has only a fitness of l but h ones. Before the modification, all schemata that lead to the global solution are superior to their competitors. Subsequently, after the modification of the mapping the fitness of all schemata h that contain y^g but not x^g, is reduced by $(h-l)/(2^{l-\lambda})$, whereas the fitness of all misleading schemata containing only x^g is increased by this amount. Schemata that contain x^g as well as y^g are not changed. As a result, the average fitness of high quality schemata is reduced, whereas the fitness of misleading schemata is increased. Let us illustrate this with a small 3-bit example. f_g from Fig. 3.18 should be modified for the genotypes 001 and 101. Therefore, $x^g = 001$ corresponds to $x^p = 101$ and $y_g = 101$ corresponds to $y_p = 001$ Then, individual $x^g = 001$ has fitness 2, and individual $y_g = 101$ has fitness 1. The fitness of the schema 1** is reduced, whereas the fitness of schema 0** increases. For size two schemata, the fitness of 10* and 1*1 decreases, whereas the fitness of 00* and 0*1 increases. As a result, the problem becomes more difficult to solve for a GA.

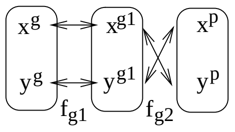

Figure 3.20. A decomposition of f_g

Finally, we could decompose f_g into two mappings f_{g1} and f_{g2} if the number of different positions in the two genotypes x^g and y^g is higher than the number of different ones $d_{x^g,y^g} > u_{x^g} - u_{y^g}$ (see Fig. 3.20). f_{g1} maps x^g to x^{g1}, and y^g to y^{g1}. x^{g1} (resp. y^{g1}) should have the same number of ones as x^g (resp. y^g) ($u_{x^{g1}} = u_{x^g}$, $u_{y^{g1}} = u_{y^g}$), but some positions are different in the two individuals x^{g1} and y^{g1} ($d_{x^{g1},y^{g1}} = u_{x^p} - u_{y^p}$). Therefore, as the number of ones stays constant, f_{g1} does not change the fitness of the schemata (compare item 1). For x^{g1} and x^p (resp. y^{g1} and y^p), the number of different ones is the same as the number of different positions. Thus, f_{g2} has the same properties as discussed in the previous item and increases the fitness of misleading schemata, as well as reduces the fitness of the high-quality schemata.

We see that most modifications of a high-locality genotype-phenotype mapping f_g make the one-max problem more difficult to solve. Only when the mapping between genotypes and phenotypes is changed that have the same number of ones in the string, is the structure of the BBs preserved, and we get the same performance as for the high-locality representation from Fig. 3.18. The above proof can be applied in the same way to a fully deceptive trap problem. Then, most of the low-locality encodings reduce the fitness of the misleading schemata, and increase the fitness of the high-quality schemata, which makes the problem easier. In the following subsection, we present an empirical verification of the results.

3.3.7 Empirical Results

In this subsection, we present an empirical investigation into how the problem complexity is changed for the one-max problem and the deceptive trap problem if low-locality representations are used. We experimentally show that for high-locality representations the fully easy one-max problem remains fully easy. In contrast, most of the low-locality representations make the one-max problem more difficult to solve for GAs. The situation is vice versa for the fully difficult deceptive trap where low-locality representations always makes the problem easier to solve.

For a non-redundant genotype-phenotype mapping f_g that is defined on binary genotypes of length l, there are $2^l!$ different possibilities to assign the 2^l genotypes to the 2^l phenotypes (Assigning the genotypes to the phenotypes can be interpreted as a permutation of 2^l numbers). Any of these possibilities represents a specific mapping like for example the binary encoding or the Gray encoding.

Table 3.6. 24 possibilities to assign four genotypes $\{a^g,\ b^g,\ c^g,\ d^g\}$ to four phenotypes $\{a^p,\ b^p,\ c^p,\ d^p\}$

x^p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
a^p	a^g	a^g	a^g	a^g	a^g	a^g	b^g	b^g	b^g	b^g	b^g	b^g	c^g	c^g	c^g	c^g	c^g	c^g	d^g	d^g	d^g	d^g	d^g	d^g
b^p	b^g	b^g	c^g	c^g	d^g	d^g	a^g	a^g	c^g	c^g	d^g	d^g	a^g	a^g	b^g	b^g	d^g	d^g	a^g	a^g	b^g	b^g	c^g	c^g
c^p	c^g	d^g	b^g	d^g	b^g	c^g	c^g	d^g	a^g	d^g	a^g	c^g	b^g	d^g	a^g	d^g	a^g	b^g	b^g	c^g	a^g	c^g	a^g	b^g
d^p	d^g	c^g	d^g	b^g	c^g	b^g	d^g	c^g	d^g	a^g	c^g	a^g	d^g	b^g	d^g	a^g	b^g	a^g	c^g	b^g	c^g	a^g	b^g	a^g

In Table 3.6, we illustrate for $l = 2$ that there are $2^2! = 24$ possibilities to assign the four genotypes $\{a^g = 00, b^g = 01, c^g = 10, d^g = 11\}$ to the four phenotypes $\{a^p = 00, b^p = 01, c^p = 10, d^p = 11\}$. Each of the 24 genotype-phenotype mappings represents a specific representation that assigns the fitness values to the genotypes in a different way and that results in a different difficulty of the problem.

In the following paragraphs, we investigate how problem difficulty changes for the one-max and deceptive trap problem if we use low-locality representations. The fitness function f_p for the fully easy l-bit one-max problem is defined as

$$f_p(x^p) = \sum_{i=0}^{l-1} x_i^p, \tag{3.24}$$

and the l-bit deceptive trap function is defined as:

$$f_p(x^p) = \begin{cases} l - 1 - \sum_{i=0}^{l-1} x_i^p & \text{for } \sum_{i=0}^{l-1} x_i^p < l, \\ l & \text{for } \sum_{i=0}^{l-1} x_i^p = l. \end{cases} \tag{3.25}$$

If we use genotypes and phenotypes of length l the number of possible representations is $2^l!$. To reduce this number, we assume without loss of generality that the phenotype x^p with only ones, which has fitness $f_p = l$, is always assigned to the individual x^g with only ones. Then, the number of different representations is reduced to $(2^l - 1)!$. For example, in Figs. 3.18 and 3.19 we have $2^l = 8$ genotypes and $2^l = 8$ phenotypes. Therefore, we have $8! = 40,340$ different representations. If we assign $x^g = 111$ always to $x^p = 111$ then there are only $7! = 5,040$ different representations. Every representation represents a different genotype-phenotype mapping.

Furthermore, we have seen in the previous subsection that for the used phenotype-fitness mapping f_p (one-max and deceptive trap problem) there are some genotype-phenotype mappings that do not modify the BBs and therefore do not change problem difficulty (both individuals have the same number of ones from item 1). These mappings have the same properties and the different f_g differ only for individuals that phenotypically have the same number of ones. There are $\prod_{i=1}^{l} \binom{l}{i}!$ representations of that kind which we want to denote as "high-locality equivalent". For example, in Fig. 3.18 we can change f_g and assign $x^g = 001$ to $x^p = 010$ and $x^g = 010$ to $x^p = 001$. Although we use a different representation, the assignment of the fitness values to the genotypes has not changed. This effect is a result of the redundancy of the used one-max and deceptive trap problem which both only consider the number of ones in the phenotype.

If we use these results we can calculate how many groups of different genotype-phenotype mappings exist that result in a different structure of the BBs. If we have $(2^l - 1)!$ different genotype-phenotype mappings and use a l-bit one-max or deceptive problem, then there are $\prod_{i=1}^{l} \binom{l}{i}!$ mappings that do not change the structure of the BBs and are equivalent to each other. Therefore, we have

$$\frac{(2^l - 1)!}{\prod_{i=1}^{l} \binom{l}{i}!}$$

groups of different genotype-fitness mappings. Each group consists of $\prod_{i=1}^{l} \binom{l}{i}!$ different genotype-phenotype mappings which do not affect the structure of the BBs and where only the mapping is changed between genotypes and phenotypes that have the same number of ones.

When using a 3-bit one-max or deceptive trap problem then there are $\frac{(2^3-1)!}{3!3!} = 5040/36 = 140$ groups of different genotype-fitness mappings with different properties. Each of these 140 different groups result in a different structure of the genotypic and phenotypic BBs. We use for our investigation 10 concatenated one-max or deceptive trap problems of size 3. Therefore, the overall string length $l = 30$, and the fitness of an individual is calculated as the sum over the fitness of the ten one-max or deceptive trap sub-problems.

To illustrate how genotype-phenotype mappings change the complexity of a problem we measure how many of the ten BBs (a BB is a correctly solved sub-problem and consists of a sequence of $l = 3$ ones) a GA finds dependent on the used representation. As we use a 3 bit problem there are 5,040 different representations and 5,040/36=140 different representations that are equivalent to each other. Only these representations that are contained in exactly one out of the 140 equivalence groups are "high-locality equivalent" ($d_m = 0$). An example is shown in Fig. 3.18. Due to the structure of the one-max and deceptive trap problem, there are 35 other representations which, although they have low locality, do not change the structure of the BBs and are "high-locality equivalent". These types of encodings assign genotypes and phenotypes with the same number of ones in a different way.

Figure 3.21 presents the results of our experiments for the 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 representations. The plot shows results for all 5,040 different representations. The ordinate counts the number of representations that allow a GA to correctly solve a certain number of sub-problems. We used a generational GA with tournament selection without replacement of size 2, uniform crossover, no mutation and a population size of $N = 15$. We performed 200 runs for each representation, and each run was stopped after the population was fully converged. The average number of correctly solved sub-problems measures the problem difficulty for the GA using one specific representation. The more sub-problems which could be correctly solved, the easier the problem is for GAs.

How can we interpret the data in Fig. 3.21? Every bar indicates the number of different representations that allow a GA to correctly identify a specific number of sub-problems. For example, the bar of height 95 at position 7.0 means that a GA correctly solves on average between 6.975 and 7.025 sub-problems for 95 different representations. The bar at position 4.85 means that there are 4 different representations that allow a GA to correctly solve on average between 4.825 and 4.875 sub-problems. The plot shows that by using a GA with only 15 individuals we solve independently of the used representation at least 4.2 sub-problems, and we are not able to correctly solve more

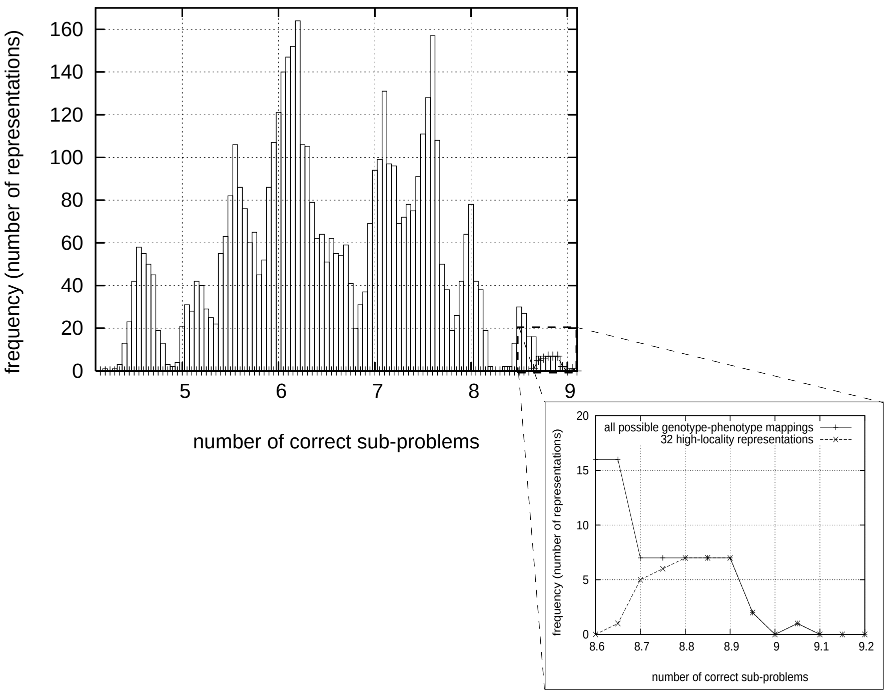

Figure 3.21. Experimental results of the frequency of the number of correct sub-problems at the end of a run for all possible encodings of a 3-bit one-max problem. We present results for 10 concatenated 3-bit problems. The optimal solution is always 111 so there are $(2^3 - 1)! = 5,040$ different representations. We use a GA with tournament selection without replacement, uniform crossover and a population size of $N = 15$. We perform 200 runs for every possible encoding. Only for these 36 representations that are equivalent to the high-locality representation, the fully easy one-max problem remains fully easy. All other encodings have low locality and make the problem more difficult to solve for GAs.

than 9 out of ten sub-problems. Furthermore, it is surprising that we have no normal distribution over the number of correct sub-problems but that there are clusters. For example, there are many representations that allow a GA to solve on average between 5.8 and 6.3 sub-problems but there are only a few representations that allow a GA to correctly solve on average between 6.5 and 6.8 sub-problems. The reason is, that there are only 140 different equivalence groups of representations. Although we have 5,040 different genotype-phenotype mappings, there are only 140 different levels of problem complexity possible. The observed clusters are probably a result of these small number of different levels of problem complexity.

It is more interesting to ask how the high-locality representation from Fig. 3.18 performs? And how the performance is of the other 35 genotype-phenotype mappings that, although they have low locality and do not preserve the distances, are equivalent to the high-locality representation and result in the same genotype-fitness mapping? The small plot in Fig. 3.21 answers these questions. The bold line shows the performance of a GA using these 36 different "high-locality-equivalent" representations. The use of these representations results in the highest GA performance. For example, there are 7 different representations that allow a GA to correctly solve between 8.825 and 8.875 sub-problems. All 7 encodings belong to the group of 36 "high-locality-equivalent" encodings. Furthermore, we see that all representations that allow a GA to correctly solve on average more than 8.75 sub-problems belong to this group. These 36 encodings result in the highest proportion of correct sub-problems. For these encodings the one-max problem remains fully easy and the size of the genotypic and phenotypic BBs stays $k_g = k_p = 1$.

Changing the representation, that means assigning the elements of Φ_g in a different way to the elements in Φ_p, always results in a low-locality representation. If the genotype-fitness mapping is not "high-locality equivalent" then the BB-complexity increases and the problem becomes more difficult to solve. A GA has more difficulties in solving the problem, and the proportion of correctly solved sub-problems is lower. The plot illustrates nicely that the used representation can change the complexity of the one-max problem dramatically. However, only encodings that are equivalent to the high-locality encoding allow a GA to efficiently solve the fully easy one-max problem.

In Fig. 3.22, we present results for 10 concatenated instances of a 3-bit deceptive trap. The GA parameters chosen are the same as for the one-max problem. The plots show that, as expected, the GA performs worst for "high-locality equivalent" representations. All other representations make the problem easier to solve for GAs.

We have empirically shown that only high-locality representations guarantee that fully easy problems remain fully easy. High-locality representations preserve BB-complexity and are a good choice if we want GAs to reliably solve problems of bounded complexity. As soon as a representation has low locality and does not preserve BB-complexity, some of the easy problems become more difficult and therefore can no longer be solved by the GA. Indeed, some of the difficult problems could become solvable by using low-locality representations, but in general we are not interested in solving these types of problems.

3.3.8 Conclusions

This section presented the third and final element of a theory of representations. We investigated how the locality of a representation influences the performance of GEAs. The locality of a representation describes how well genotypic neighbors correspond to phenotypic neighbors. It is high if genotypic neighbors correspond to phenotypic neighbors. The results show that high-locality representations preserve the difficulty of a problem and phenotypically easy problems also remain genotypically easy. Using low-locality

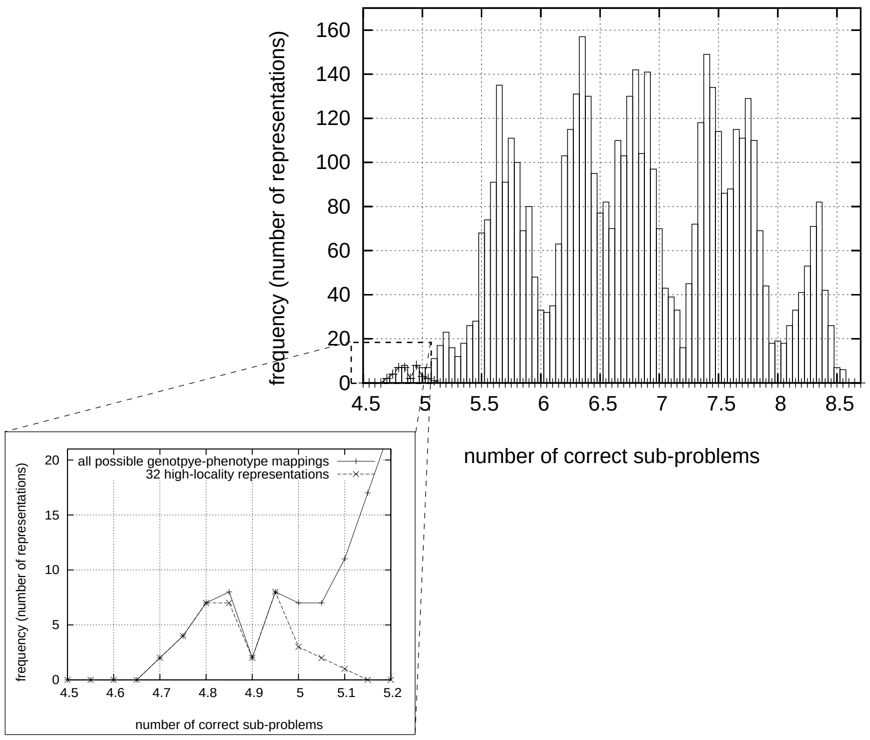

Figure 3.22. Experimental results of the frequency of the number of correctly solved sub-problems at the end of the run for all possible encodings of a 3-bit deceptive-max problem. We present results for 10 concatenated 3-bit problems. The optimal solution is always 111 so there are $(2^3 - 1)! = 5,040$ different possible encodings. The GA uses tournament selection without replacement, uniform crossover and a population size of $N = 15$. We perform 200 runs for every possible encoding. Only for these 36 representations that are equivalent to the high-locality encoding does the fully difficult deceptive trap remain fully difficult. For all other representations, the BB-complexity is reduced and the problem becomes easier to solve for GAs.

representations is equivalent to randomizing the search process. Therefore, low-locality representations change problem difficulty and make easy problems more difficult and deceptive problems more easy to solve.

In general, we want GEAs to be able to solve a class of problems of bounded complexity fast and reliably. However, the results have shown that the use of low-locality representations in general changes problem difficulty and can only increase problem difficulty for fully easy problems. Therefore, easy problems that are solvable using a high-locality representation could become unsolvable when using a low-locality representation that modifies BB-complexity. To guarantee that a GA can reliably solve problems of bounded complexity it is designed for, we recommend the use of high-locality representations.

This section nicely illustrated that representations can dramatically change the complexity of a problem. The presented work has shown that even fully difficult problems can be solved easily if a proper, low-locality, representation is used. However, using the same representation for a fully easy one-max problem can make the problem fully difficult and unsolvable. Therefore, the use of low-locality representations could be advantageous if we know that the problem is deceptive. But in general, users do not have this information and therefore, they should not use these types of representations.

3.4 Summary and Conclusions

In Sect. 3.1 we described, analyzed, and modeled the effect of redundant representations on the performance of GEAs. We distinguished between synonymously and non-synonymously redundant representations and illustrated that non-synonymous redundancy does not allow genetic operators to work properly and therefore reduces the efficiency of evolutionary search. For synonymously redundant representations, GEA 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 performance 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 there exists a-priori some information about the optimal solution.

This was followed in Sect. 3.2 by an investigation into how the scaling of an encoding influences the performance of GEAs. We extended previous work (Rudnick 1992; Thierens 1995; Thierens et al. 1998; Harik et al. 1997) and formulated a more exact convergence model considering genetic drift for exponentially scaled representations. Representations are exponentially scaled if the contribution of the genotypic alleles to the construction of the phenotypic alleles is exponentially different. Using the developed population sizing model, we were able to more accurately predict the behavior of GEAs using exponentially scaled representations.

Finally, we presented the third and final element of a theory of representations, namely the influence of locality on problem complexity. In Sect. 3.3, we showed that high-locality representations, which preserve the neighborhood structure when mapping genotypes to phenotypes, do not modify the difficulty of a problem. When using low-locality representations, on average problem difficulty changes. On average, fully easy problems become more difficult, and deceptive problems easier. We have discussed why representations that keep easy problems easy and make deceptive problems easier are nice to have, but not possible without having an exact knowledge about the optimization problem a priori.

In this chapter, we identified three important elements towards a general theory of representations. We identified redundancy, scaling, and locality/distance distortion as having a major influence on the performance of GEAs. We were able to show that synonymously redundant encodings do not modify the performance of a GEA as long as the representation is uniformly redundant. Our investigation into non-uniformly scaled representations has shown that these types of encodings prolong the search process and increase the problems of GEAs with genetic drift. Finally, we have seen that low-locality representations do not preserve BB-complexity in general and make phenotypically easy problems more difficult. Therefore, to make sure that GAs are able to reliably solve easy problems and problems of bounded complexity, the use of high-locality representations is recommended.

Even by only presenting some basic elements of a general theory of representations we are able to analyze and predict the behavior and performance of GEAs using existing representations significantly better. The presented theory gives us a deeper understanding on how existing representations influence the performance of GEAs, as well as allows us to design new representations in a more theory-guided way. By using the presented theory, on the one hand we can develop general and robust representations that can be applied to problems of unknown complexity, and on the other hand problem-specific representations which could fail for some problems, but perform well for a specific problem.

Although the provided elements of representation theory already allow a guided design and analysis of representations, further research is still necessary to develop a general representation theory. Especially, the relationship between the presented elements of theory of representations should be investigated more deeply. We believe that as we are able to easily separate the effects of redundant and exponentially scaled representations that there is not much interconnection and overlapping between these two elements of theory. However, for locality and its influence on BB-complexity, the situation is different. We have seen that the modification of problem complexity is strongly influenced by redundancy or scaling. Therefore, further research is necessary to identify the exact relations between the presented elements of theory.

Finally, we want to encourage researchers to do more basic research towards the development of a general theory of representations. We believe that we provided some important parts, but there is still a long way to go. However, the path is worth following, as a general theory of representations would allow us to unleash the full power of genetic and evolutionary search and help us to solve problems fast, accurately and reliably.