On the Locality of Representations

Franz Rothlauf

Working Paper 1/2003
January 2003

Working Papers in Information Systems 1

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

On the Locality of Representations

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

January 20, 2003

Abstract

It is well known that using high-locality representations is important for efficient evolutionary search. This paper discusses in detail how the locality of a representation influences the difficulty of a problem when using mutation-based search approaches. The results show that high-locality representations do not change problem difficulty. In contrast, low-locality representations randomize the search process and make problems that are easy for mutation-based search more difficult and difficult problems more easy. Although low-locality representations increase the performance of local search on difficult, deceptive problems this is not relevant for real-world problems as we assume that most problems in the real-world are easy for mutation-based search.

1 Introduction

The locality of a representation describes how well genotypic neighbors correspond to phenotypic neighbors. Previous work has indicated that high locality of a representation is necessary for efficient evolutionary search [1, 2, 3, 4]. However, it remains unclear as to how the locality of a representation influences problem difficulty in general, and if high-locality representations always increase the performance of evolutionary search.

The purpose of this paper is to investigate how the locality of a representation influences problem difficulty when using mutation as the main search operator. Furthermore, it discusses how the locality of a representation depends on the used search operator and on the metric that is defined on the phenotypic and genotypic search space. The results show that when using high-locality representations the phenotypic and genotypic problem difficulty remain the same. The difficulty of optimization problems does not change. Therefore, easy real-world problems can be solved by evolutionary algorithms (EA) more efficiently when using high-locality representations. In contrast, low-locality representations change problem difficulty. Such representations increase the difficulty of problems which are easy for mutation-based search and reduce the difficulty of problems that mislead mutation-based search.

The paper is structured as follows. In the following section we introduce genotypes and phenotypes, introduce the locality of a representation, and discuss how mutation operators are based on the metric that is defined on the search space. Section 3 focuses on the phenotype-fitness mapping. It reviews some common measurements of problem difficulty and presents a classification of problem difficulty which is based on the work of [5]. In section 4 we investigate how genotype-phenotype

mappings change problem difficulty when assigning genotypes to phenotypes and when using mutation as the main search operator. It distinguishes between low and high-locality representations and illustrates that when using a high-locality representation genotypic and phenotypic problem difficulty are the same. In contrast, problem difficulty changes if low-locality representations are used. The paper ends with concluding remarks.

2 Representations, Locality and Mutation Operators

The following section presents the prerequisites for our investigation. It decomposes the optimization problem into a genotype-phenotype and a phenotype-fitness mapping, defines the locality of a representation and discusses some properties of mutation-based search.

2.1 Genotypes, Phenotypes, and Fitness

When using some kind of representation, every optimization problem f can be decomposed into a genotype-phenotype mapping f_g, and a phenotype-fitness mapping f_p [6].

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 genotypic space Φ_g. The optimization problem is defined by finding the unique global 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 a function which assigns a fitness value to every $\boldsymbol{x}$. The vector $\hat{\boldsymbol{x}}$ is the global maximum.

When using a representation it must be distinguished between phenotypes and genotypes [7]. Thus, the optimization problem 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 [8, 9].

2.2 Metrics and Locality

In the following subsection we 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 $d_{\boldsymbol{x}_a,\boldsymbol{x}_b}$ between two individuals $\boldsymbol{x}_a \in \Phi$ and $\boldsymbol{x}_b \in \Phi$ 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 [10] for binary strings the minimal distance between two individuals is $d = 1$. Therefore, two individuals $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ are neighbors if the distance $d_{\boldsymbol{x}_a,\boldsymbol{x}_b} = 1$.

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 $x_1^p = 5$ has two neighbors, $x_2^p = 6$ and $x_3^p = 4$. When using the Hamming metric, the corresponding binary string $x_1^g = 101$ has three different neighbors, $x_2^g = 001$, $x_3^g = 111$, and $x_4^g = 100$ [11].

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

$$d_m = \sum_{d^p_{\boldsymbol{x}_i,\boldsymbol{x}_j}=d^p_{min}} |d^g_{\boldsymbol{x}_i,\boldsymbol{x}_j} - d^g_{min}|,$$

where $d^p_{\boldsymbol{x}_i,\boldsymbol{x}_j}$ is the phenotypic distance between the phenotypes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $d^g_{\boldsymbol{x}_i,\boldsymbol{x}_j}$ 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 locality.

We want to emphasize that the locality of a representation depends not only 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 talk about the locality of a representation a metric must be defined on Φ_g and Φ_p.

2.3 Mutation-based Search

In the following subsection we briefly discuss how mutation operators are based on the metric that is used for the genotypic space Φ_g.

Based on the metric defined on the genotypic search space Φ_g, search operators like mutation 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) [12]. 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, only 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 [13] 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 (compare [14]). 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.

3 Phenotype-Fitness Mappings

To be able to investigate how the locality of a representation influences the difficulty of problems for mutation-based search we summarize in subsection 3.1 some existing measurements of problem difficulty. Consequently, we review briefly in subsection 3.2 a classification of problem difficulty which we want to use in section 4 for our investigations.

As described in subsection 2.1 the difficulty of a problem depends on the phenotype-fitness mapping f_p. f_p determines the fitness of the solutions. Furthermore, problem difficulty depends on 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. 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.

In subsection 2.3 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 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.

This aspect of problem difficulty can be illustrated by examining problem difficulty when using random search. When using random search it can not be distinguished between easy and difficult problems. The search process chooses new individuals randomly and all possible problem instances of the same problem and size have the same difficulty. Although measurements of problem difficulty may lead us to believe that some problem instances are easier than others, there are no easy or difficult problems when using random search. Independently of the specific structure of a problem, random search shows the same performance for problems of the same size (if we assume only one global optimal solution).

Therefore, we should bear in mind that measurements of problem difficulty always measure the difficulty of a problem regarding to a specific search method. There is no absolute measure of problem difficulty.

3.1 Measurements of Problem Difficulty

In the following subsection we shortly summarize some common measurements of problem difficulty and describe for which type of search operators they are most appropriate. Common measurements of problem difficulty are:

- Correlation analysis,
- polynomial decomposition and Walsh coefficients, and
- schemata analysis.

These measurements of problem difficulty are widely used in the EA literature for measuring problem difficulty [15, 16, 17, 18, 5]. For an overview see [19]. Their specific properties are briefly discussed in the following paragraphs.

Correlation analysis is based on the assumption that the high and low-quality solutions are grouped together and neighboring individuals have similar fitnesses. Problems are easy if the structure of the search space guides the search to the high-quality solutions. Consequently, correlation analysis is a proper measurement of problem difficulty when using mutation-based search approaches. The most common measurements for distance correlation are the autocorrelation function of the fitness landscape [20, 13], the fitness correlation coefficients of genetic operators [13], and the fitness-distance correlation [21, 5, 22].

The linearity of an optimization problem can be measured by the polynomial decomposition of the problem [23]. The polynomial coefficients describe the non-linearity of the problem. If there are high-order coefficients in the decomposition of the problem, the function is highly non-linear. If the decomposition of a problem has only order 1 coefficients, then the problem is linear and easy for most EAs. The polynomial decomposition and the Walsh decomposition are equivalent to each other as it can be shown that Walsh coefficients are also polynomials [24, 23]. For further information about Walsh analysis we refer the reader to [24, 25]. Similar to the polynomial decomposition, problems are easy for EAs if the Walsh coefficients are of order one [24, 26, 16, 27].

The analysis of the schemata is a proper way to measure problem difficulty when using crossover as the main search operator for binary problem domains [28, 29]. It is assumed that schemata and building blocks are processed and the difficulty of a problem (intra-BB difficulty [30]) can be measured by the maximum length δ and size k of the BBs [29]. A problem is denoted to be deceptive of order k_{max} if for $k < k_{max}$ all schemata that contain parts of the best solution have lower fitness than their competitors [31, 32]. Schemata are competitors if they have the same fixed positions. Therefore, the highest order k_{max} of the schemata that are not misleading determines problem difficulty for crossover-based genetic algorithms used for binary problems. The higher the maximum order k_{max} of the schemata, the more difficult a problem is.

3.2 Classification of Problem Difficulty

In the following subsection we describe in more detail the classification of problem difficulty we use for our investigation which is based on the work of [5].

As already discussed, 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 [21, 5]. 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.

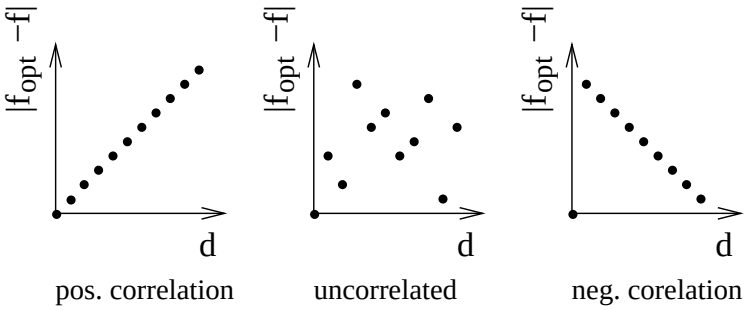

Figure 1: Different classes of problem difficulty

The three different classes of problem difficulty are illustrated in Figure 1. We show how the fitness difference $|f_{opt} - f|$ depends on the distance to the optimal solution d. In the following paragraphs we want to discuss these three classes in some more detail.

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 behaviour. Such problems are easy for local 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 approriate 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 emphazise 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 as well as random search, and for problems of class three random search performs even better. This situation is not in constrast 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. In general, for real-world problems the fitness values of neighboring solutions

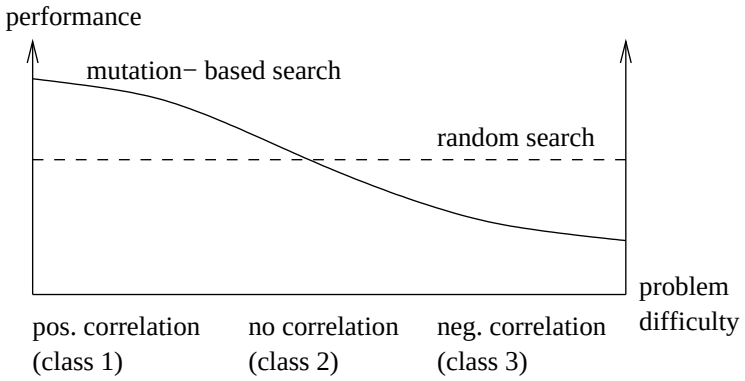

Figure 2: Performance of mutation-based EA search versus random search

are correlated, and high-quality and low-quality solutions are grouped together. Fitness landscapes that are uncorrelated, or even deceptive, are uncommon in real world.

This situation is illustrated in Figure 2. We know from the no-free-lunch theorem that all search methods show on average the same performance over all possible problem instances [33, 34]. 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 solved using mutation-based EAs.

4 Genotype-Phenotype Mappings

We have seen in section 2.1 that representations can change the character and difficulty of optimization problems. If $f(\boldsymbol{x_g}) \neq f_p(\boldsymbol{x_p})$, the genotypic and phenotypic problem difficulty is different. In the following subsections we discuss high versus low-locality representations and illustrate how the locality of a representation influences problem difficulty.

4.1 Low versus High-Locality Representations

We have seen in subsection 2.2 that the metric defined on Φ_p can be different from the metric defined on Φ_g. The locality of a representation describes 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 constrast, representations have low locality if neighboring genotypes do not correspond to neighboring phenotypes. Figure 3 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 the distances between the phenotypes are the same as the distances between the corresponding genotypes. Then a mutation step has the same effect in the phenotypic and genotypic search space.

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 assignes 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.

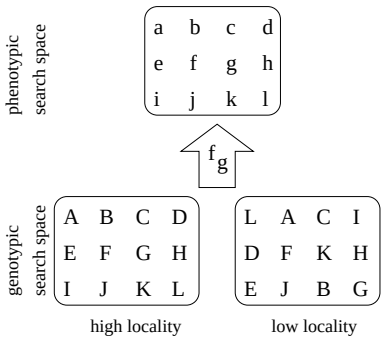

Figure 3: High versus low-locality representations

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 [35]. 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 [36]. Both metric-representation combinations result in the same fitness landscape and therefore in the same performance of mutation-based search.

4.2 Influence on Problem Difficulty

Finally, we want to discuss in the following paragraphs how the low versus high locality of a representation influences the performance of mutation-based search approaches. A representation transforms the phenotypic problem f_p with some given phenotypic problem difficulty into a genotypic problem $f = f_p \circ f_g$ with a resulting problem difficulty that can be different from the phenotypic problem difficulty. We use the problem classification described in subsection 3.2.

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 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, also the character of the search operator can be different for genotypes and phenotypes. If the locality of a representation is high, then the mutation 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. Phenotypic easy problems remain genotypically easy and phenotypic difficult problems remain genotypically difficult. Figure 4 (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. Then the influence of the representation on the difficulty of a problem depends on the considered optimization problem. If the considered problem f_p is easy (class 1) and the structure of the search space guides the mutation-based 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 of a genotype does not correspond to a small change of the phenotype but larger changes of the phenotype are possible (compare Figure 4 (right)). Therefore, when using low-locality representations, phenotypic easy problems of class one turn on average into 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 individuum 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 contant as random search and mutation-based search show the same performance for problems of class two. Of course, there are representations that can make a problem easier and result in an overall genotypic problem f of class one; however, there are only few of them and most of the low-locality representations just modify the problem and do not create a fitness landscape of class one that leads the search method to the good solutions. On the other hand, there are also representations f_g that construct a problem f that 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 of such representations.

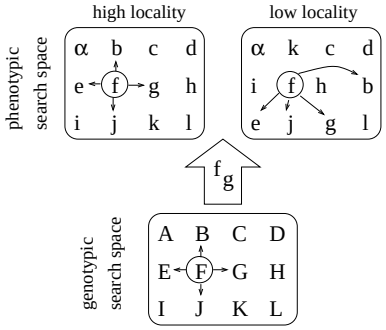

Figure 4: The effect of mutation for high versus low-locality representations

Finally, we have to consider problems of class three. On average, the use of low-locality representations transform such problems into problems of class two as the problems become more randomized. Therefore, 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.

A good example for a low locality representation are Prüfer numbers [37]. This representation represents a tree by a sequence of integers and has low locality. Much work has investigated the influence of Prüfer numbers on EA (for a survey see [38, section 6.2]) and it has been noted that most real-world problems become more difficult to solve for mutation-based EAs when using Prüfer numbers. This situation can be explained as most real-world tree problems belong to class one. Therefore, the low locality of the Prüfer number representation randomizes these problems and makes them more like class two problems which are much more difficult to solve for mutation-based EAs. [38, section 8.1.] investigated the performance of EAs using Prüfer numbers for deceptive problems (class three). The results show that such difficult problems become easier to solve and EAs using Prüfer numbers show good performance in comparison to other representations. In this case, difficult problems of class three are transformed into easier but still difficult problems of class two.

Summarizing the results we recognize that low-locality representations have the same effect as 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 which are only of theoretical interest.

5 Conclusions

This paper investigated how the locality of a represenation influences the performance of mutation-based search. The locality of a representation describes how well the metric defined on the genotypic search space corresponds to the metric defined on the phenotypic search space. For our investigation we could use many existing results and insights about representations and problem difficulty; it was not neccessary to develop much new theory but just to put the pieces into the correct order.

The results show that representations can change the difficulty of problems. Only when using

high-locality representations problem difficulty does not change. Easy problems remain easy and difficult problems remain difficult for mutation-based search. In contrast, low-locality representations randomize the search process and reduce the correlation between the fitness of a solution and its distance to the optimal solution. Therefore, problem difficulty changes and easy problems which guide mutation-based search to good solutions become more difficult and difficult problems that mislead mutation become more easy.

We strongly encourage users to use high-locality representations for real-world problems. Such representations ensure that easy problems remain easy for mutation-based search. Although, low-locality representations make deceptive problems easier, low-locality representations are not a good choice as deceptive problems are not common in the real-world.

References

[1] J. Gottlieb and G. Raidl. Characterizing locality in decoder-based eas for the multidimensional knapsack problem. In C. Fonlupt, J.-K. Hao, E. Lutton, E. Ronald, and M. Schoenauer, editors, *Proceedings of Artificial Evolution*, volume 1829 of *Lecture Notes in Computer Science*, pages 38–52. Springer, 1999.

[2] Jens Gottlieb and Günther R. Raidl. The effects of locality on the dynamics of decoder-based evolutionary search. In D. Whitley, D. E. Goldberg, E. Cantú-Paz, L. Spector, L. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference 2000*, pages 283–290, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

[3] Jens Gottlieb, Bryant A. Julstrom, Günther R. Raidl, and Franz Rothlauf. Prüfer numbers: A poor representation of spanning trees for evolutionary search. IlliGAL Report No. 2001001, University of Illinois at Urbana-Champaign, Urbana, 2001.

[4] F. Rothlauf and D. E. Goldberg. Prüfernumbers and genetic algorithms: A lesson on how the low locality of an encoding can harm the performance of GAs. In Schoenauer et al. [40], pages 395–404.

[5] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. Eschelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, 1995. Morgan Kaufmann.

[6] G. E. Liepins and M. D. Vose. Representational issues in genetic optimization. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:101–115, 1990.

[7] R. C. Lewontin. *The Genetic Basis of Evolutionary Change*. Number 25 in Columbia biological series. Columbia University Press, New York, 1974.

[8] J. D. Bagley. *The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms*. PhD thesis, University of Michigan, 1967. (University Microfilms No. 68-7556).

[9] M. D. Vose. Modeling simple genetic algorithms. In Whitley [39], pages 63–73.

[10] R. Hamming. *Coding and Information Theory*. Prentice-Hall, 1980.

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

[12] J. Doran and D. Michie. Experiments with the graph traverser program. *Proceedings of the Royal Society of London (A)*, 294:235–259, 1966.

[13] B. Manderick, M. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150, San Mateo, CA, 1991. Morgan Kaufmann.

[14] D. Whitley. Evaluating evolutionary algorithms. Tutorial Program at Parallel Problem Solving from Nature (PPSN 2002, September 2002.

[15] D. E. Goldberg. Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems*, 3:153–171, 1989.

[16] D. E. Goldberg. Construction of high-order deceptive functions using low-order Walsh coefficients. *Annals of Mathematics and Artificial Intelligence*, 5:35–48, 1992.

[17] N. J. Radcliffe. Genetic set recombination. In Whitley [39], pages 203–219.

[18] J. Horn. Genetic algorithms, problem difficulty, and the modality of fitness landscapes. IlliGAL Report No. 95004, University of Illinois at Urbana-Champaign, Urbana, IL, 1995.

[19] K. Deb, L. Altenberg, B. Manderick, T. Bäck, Z. Michalewicz, M. Mitchell, and S. Forrest. Theoretical foundations and properties of evolutionary computations: fitness landscapes. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages B2.7:1–B2.7:25. Institute of Physics Publishing and Oxford University Press, Bristol and New York, 1997.

[20] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336, 1990.

[21] T. Jones. *Evolutionary algorithms and heuristic search.* Unpublished doctoral dissertation, University of New Mexico, Alberquerque, NM, 1995.

[22] L. Altenberg. Fitness distance correlation analysis: An instructive counterexample. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 57–64, San Francisco, 1997. Morgan Kaufmann.

[23] G. E. Liepins and M. D. Vose. Polynomials, basis sets, and deceptiveness in genetic algorithms. *Complex Systems*, 5(1):45–61, 1991.

[24] D. E. Goldberg. Genetic algorithms and Walsh functions: Part I, a gentle introduction. *Complex Systems*, 3(2):129–152, 1989.

[25] M. D. Vose and A. H. Wright. The simple genetic algorithm and the Walsh transform: Part I, theory. *Evolutionary Computation*, 6(3):253–273, 1998.

[26] C. K. Oei. Walsh function analysis of genetic algorithms of non-binary strings. Master's thesis, University of Illinois at Urbana-Champaign, Department of Computer Science, Urbana, 1992.

[27] C. Reeves and C. Wright. An experimental design perspective on genetic algorithms. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 7–22, San Francisco, California, 1994. Morgan Kaufmann Publishers, Inc.

[28] J. H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[29] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA, 1989.

[30] David E. Goldberg. *The Design of Innovation.* Series on Genetic Algorithms and Evolutionary Computation. Kluwer, Dordrecht, The Netherlands, 2002.

[31] D. E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. In *Genetic Algorithms and Simulated Annealing*, chapter 6, pages 74–88. Morgan Kaufmann, San Mateo, CA, 1987.

[32] K. Deb and D. E. Goldberg. Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, 10:385–408, 1994.

[33] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Tech. Rep. No. SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, 1995.

[34] D. Whitley. Functions as permutations: Implications for no free lunch, walsh analysis and statistics. In Schoenauer et al. [40], pages 169–178.

[35] C. R. Reeves. Landscapes, operators and heuristic search. *Annals of Operational Research*, 86:473–490, 1999.

[36] C. Reeves. Fitness landscapes: A guided tour. Joint tutorials of SAB 2000 and PPSN 2000, tutorial handbood, 2000.

[37] H. Prüfer. Neuer Beweis eines Satzes über Permutationen. *Archiv für Mathematik und Physik*, 27:742–744, 1918.

[38] Franz Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Studies on Soft Computing and Fuzziness. Springer Verlag, Berlin, 2002.

[39] L. D. Whitley, editor. *Foundations of Genetic Algorithms 2*, San Mateo, CA, 1993. Morgan Kaufmann.

[40] M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors. *Parallel Problem Solving from Nature, PPSN VI*, Berlin, 2000. Springer-Verlag.