Centre for Informatics and Systems of the University of Coimbra

Technical Report 006 September 2007

Golomb Rulers: the Influence of Representation and Heuristics

Jorge Tavares jast@dei.uc.pt
Centre for Informatics and Systems of the University of Coimbra
Polo II - Pinhal de Marrocos
3030 Coimbra, Portugal

Francisco B. Pereira xico@dei.uc.pt
Polytechnic Institute of Coimbra
Quinta da Nora
3030 Coimbra, Portugal

Ernesto Costa ernesto@dei.uc.pt
Centre for Informatics and Systems of the University of Coimbra
Polo II - Pinhal de Marrocos
3030 Coimbra, Portugal

Abstract

Fitness landscape analysis techniques are used to better understand the influence of genetic representations and associated variation operators when solving a combinatorial optimization problem.

Several representations for the Optimal Golomb Ruler problem are examined. Common mutation operators such as bit-flip mutation and classic one point and uniform crossover are employed to generate fitness landscapes to study the genetic representations. Furthermore, additional experiments are made to observe the effects of adding heuristics and local improvements to the encodings.

The fitness landscape analysis was complemented with a statistical analysis. Several hypothesis tests were used in order to verify and support the previous findings. The research is conclued by relating the analysis with results obtained from optimization runs and additional experimentation concerning insertion and correction procedures.

Keywords

Representation, heuristics, fitness landscapes, mutation, crossover, heuristic bias, locality, local improvement, Golomb rulers.

 ISSN 0874-338X

1 Introduction

Combinatorial optimization problems (Papadimitriou and Steiglitz, 1982) emerge in many diverse fields like engineering, biology, chemistry, physics, management and computer science, with numerous practical applications like transportation and logistics, network routing, VLSI design and protein folding to mention just a few examples. For all these problems, the task is to find the best solution, the optimum, from the set of all possible solutions. A characteristic of this class of problems, contrary to other optimization problems, is that they have a finite number of candidate solutions. In spite of that, they are known to be very complex and therefore, hard to solve. They belong to the NP complexity class.

As such, for the majority of the cases where the computation complexity is high, the best way to tackle combinatorial optimization problems is to apply heuristics methods. Although there is not a guarantee of finding the best solution to a problem, most of the times these approaches converge quickly to high quality solutions. For most situations, these high quality solutions are good enough for practical use. Thus, researchers have put a considerable amount of effort in developing and studying these heuristic methods.

Evolutionary algorithms (Eiben and Smith, 2003) are computational procedures inspired in the principles of natural selection of Darwin and by the genetics of Mendel, integrating a set of innovative techniques used for solving complex problems. These algorithms have proved to be an efficient and robust paradigm for problem solving. Due to their probabilistic nature and to the fact of making an implicit parallel space search, they can, for the most part of the cases, find solutions of good quality in an acceptable time

One of the fundamental questions in the development of evolutionary algorithms is the representation of candidate solutions for a given problem. Part of the success of this type of approaches relies on the fact of obtaining one representation that allows the algorithm to perform the most efficient and accurate search through the search space. The representation must be a) complete, i.e., all possible candidate solutions to the problem must be encoded; and b) with bias towards fitter solutions. It should also be easy to manipulate them by means of the variation operators. According to the type of problem, this can be attained in different ways not only from the traditional binary or integer alphabets, but also with the inclusion of specific knowledge about the problem at hand. This fact raises a set of pertinent questions during the development of an evolutionary algorithm: What is the most suitable representation for a problem? Which representation gives more guarantees of discovering solutions with better quality or even optimal?Is it possible to arrive more quickly to a solution, i.e., does a representation allow an efficient exploration of the search space? Does a particular representation need auxiliary mechanisms, e.g., heuristics, in order to achieve efficiency and/or accuracy? Which of these complementary procedures are influential? In which way is the representation really important?

The study of representations and operators in evolutionary algorithms is a recurrent topic in the area of the evolutionary computation. However, most of the current studies cover the application of these components to new problems. In general, the emphasis of the investigations is focused on demonstrating how an evolutionary algorithm is capable of attaining good solutions for a given problem and not in the reasons why good solutions can be discovered. In fact, there is not a sound theory of representations capable to explain and to help researchers in developing and studying the behavior of an evolutionary algorithm for a specific problem in relation with the choice

of the representation and associated operators. In spite of that, researchers have developed some tools that can help them in the hard task of studying and analyzing representations with their associated operators and complementary procedures. Fitness landscapes (Wright, 1932) illustrate the association between the search space and the fitness space. An evolutionary algorithm can be seen as navigating a landscape in order to find the highest peak. Higher points in the search space correspond to solutions with higher fitness[1]. A representation and associated variation operators influence the search efficiency of an evolutionary algorithm. With this in mind, the choice of a representation and operators can be made based on the study of the difficulties of the corresponding fitness landscapes. Since fitness landscape analysis techniques were introduced they have become a valuable tool to investigate why a given evolutionary algorithm works.

Golomb Rulers are numerical sequences, or a class of undirected graphs, named after the relevant work of the mathematician Solomon Golomb (Golomb, 1972), (Bloom and Golomb, 1977), that find application in a variety of engineering fields (Rankin, 1993). A good example is in the field of communications. When setting up an interferometer for radio astronomy, where placing the antennas on the marks of a Golomb ruler maximizes the recovery of information about the phases of the signal received (Blum et al., 1974), (Hayes, 1998). Other examples of areas where they are used include X-ray crystallography (Bloom and Golomb, 1977), or error detection and correction in the field of coding theory (Dollas et al., 1998), (Klove, 1989).

In this work we present our investigations towards a better understanding of the role of representation and genetic operators when evolutionary algorithms are applied to the OGR. There is a large selection of representations, which implies that a careful analysis of representation, variation operators and other techniques, such as the use of heuristics and local optimization, is essential to the design of an effective algorithm for this problem.

The paper is structured as follows: a description and formal definition of the problem is presented in section 2; section 3 presents an overview of evolutionary techniques applied to the OGR; in section 4 we explain the concept of fitness landscapes and describe the analysis techniques used for this work; experimental results and discussion are reported in section 5; finally, in section 6 we present some conclusions.

2 Problem Description

A Golomb Ruler is defined as a ruler that has marks unevenly spaced at integer locations in such a way that the distance between any two marks is unique. Unlike usual rulers, they have the ability to measure more discrete measures than the number of marks they possess. Additionally, Golomb Rulers are not redundant since they do not measure the same distance twice. Although the definition of a Golomb Ruler does not place any restriction on the length of the ruler, researchers are usually interested in rulers with minimum length.

An Optimal Golomb Ruler (OGR) is defined as the shortest length ruler for a given number of marks (Dollas et al., 1998). There may exist multiple different OGRs for a specific number of marks. A Perfect Golomb Ruler is a particular case of an OGR, since in addition to the minimum length requirement, it should measure all distances between 1 and the overall length of the ruler. In figure 1, a Perfect Golomb Ruler with 4 marks is presented.

[1] In a maximization problem. It is also trivial to apply to minimization problems.

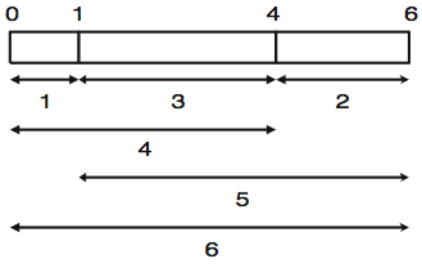

Figure 1: A Perfect Golomb Ruler with 4 marks.

As can be seen, all distances between 1 and 6 (the length of the ruler) can be measured. It can be proved that perfect rulers beyond 4 marks do not exist (Dollas et al., 1998). For a small number of marks it is possible to construct OGRs by hand. As the number of marks increases, the problem becomes more difficult and, for $n \geq 9$, computational approaches are required to find possible solutions. Currently, most of the techniques used to identify OGRs rely on massive parallel brute force algorithms. Since this is a NP-hard problem (Rankin, 1993), computation is prohibitively costly in terms of computer resources.

A formal definition of Golomb Rulers can be stated as: an n-mark Golomb Ruler is an ordered set of n distinct nonnegative integers $\{a_1, a_2, \ldots, a_n\}$ such that all possible differences $|a_i - a_j|$, $i, j = 1, \ldots, n$ with $i \neq j$, are distinct. Values a_i correspond to positions where marks are placed. By convention, the first mark a_1 is placed in position 0, whereas the length of the ruler is given by the position of the rightmost mark a_n. For example, the ruler from figure 1 can be defined as $\{0, 1, 4, 6\}$.

The length of a segment of a ruler is defined as the distance between two consecutive marks. Thus, it is also possible to represent a Golomb Ruler with n marks through the specification of the length of the $n - 1$ segments that compose it. According to this notation the example from figure 1 can be defined as $\{1, 3, 2\}$. The Golomb Ruler $\{a_1, a_2, \ldots, a_n\}$ is an OGR if there is no other n-mark ruler with a smaller largest mark, a_n. In such a case a_n is called the length of the n-mark OGR (abbreviated OGR-n).

Given its interest, both for real world applications and as a mathematical puzzle, the search for OGRs has attracted the attention of researchers over the past few decades. Several works regarding Golomb Rulers can be found in the literature. We present a brief description of some of the classical approaches for this problem. After that, we will present and discuss the evolutionary approaches.

One of the most important classes of algorithms proposed aims to construct OGRs from scratch. (Dewdney, 1985) presented one of these methods. The proposed technique is composed of two phases: the ruler generation and the Golomb ruler verification. The generation phase takes two parameters as input (the number of marks and an upper bound on the ruler length) and recursively tries to construct a Golomb ruler. The second phase of the algorithm verifies if the generated solution satisfies all requirements for a Golomb ruler. (Gardner, 1972) and (Dollas et al., 1998) presented other examples of algorithms that iteratively construct solutions. In order to improve execution time, most of these methods consider a set of search space reduction techniques.

Shearer (1990) proposed a depth first backtracking algorithm. During the search this method builds a Golomb ruler by selecting the position of the marks in order. At

each node of the search tree the program keeps track of which differences have been used and which integers could be adjoined to the ruler without leading to an invalid solution. The algorithm backtracks when too few integers remain eligible to allow a completion of the ruler.

Best solutions for OGRs with a number of marks between 20 and 23 were obtained by massive parallel distributed projects. Two projects are involved in searching for OGRs: GVANT[2] and distributed.net[3]. Both of them are collective computing projects that rely on spare CPU cycles from a large number of computers distributed over the Internet. It took several months and numerous collaborators to find optimum solutions for each one of the instances with 20, 21, 22, 23 and 24 marks. The search for 25 marks started in July 2000 and is still running.

3 Evolutionary Approaches

Contrary to other optimization problems, evolutionary algorithms have not been widely applied to the OGR. To the best of our knowledge, there were few attempts to apply evolutionary techniques to search for OGRs.

The first work regarding the application of evolutionary algorithms to Golomb Rulers was the approach proposed by (Soliday et al., 1995). This work consisted of a standard evolutionary algorithm with simple modifications to allow the search for OGRs and for a long time the proposed algorithm was the only application of evolutionary techniques to this problem. It adopted a representation where an OGR candidate is a permutation of segments, built from two lists containing segments of the ruler. Special precautions were taken during the generation of the initial population to ensure that an individual did not contain the same segment twice. Moreover, the genetic operators were also required to guarantee that descendants did not have repeated segments. The results achieved were not very good: best solutions evolved for rulers with 10 to 16 marks are far from the known overall best. Latter, in section 5.6 we present table 9 comparing the results attained by the approaches.

Feeney (2003) proposed and studied a hybrid approach to search for Golomb rulers. His work was a continuation of (Soliday et al., 1995) approach and consisted in applying local improvement techniques with evolutionary algorithms. Feeney investigated three local search methods coupled with Baldwinian and Lamarckian learning. It is known from previous studies that the addition of learning to evolutionary algorithms can, under some circumstances, help the evolutionary search process in finding a near-optimum solution (Pereira, 2002). Nevertheless, learning has also a drawback since it can result in a premature convergence to suboptimal solutions. Achieved results were an improvement to the ones attained by (Soliday et al., 1995).

The fact that few applications of evolutionary algorithms to this problem existed has led us to develop two different approaches for finding OGRs: (Pereira et al., 2003) and (Tavares et al., 2004a). At the same time, this problem allowed us to use it as a benchmark case to test and study genetic representations usually found in combinatorial optimization problems. Our first approach (Pereira et al., 2003) used an evolutionary algorithm with a decoder-based representation. The candidate solution is a permutation of segments encoded with random-keys based on the NetKeys extension proposed by (Rothlauf et al., 2002). That way we could use standard genetic operators. The second proposed evolutionary algorithm,(Tavares et al., 2004a) followed a different strategy. It uses binary representation to represent a candidate solution. In this partic-

[2]GVANT project. Available at http://members.aol.com/golomb20/
[3]Distributed.net "Project OGR". Available at http://www.distributed.net/ogr/

ular case, the candidate solution represents the position of marks in a Golomb Ruler instead of the segments length. This algorithm was also augmented with additional techniques, such as heuristics and local improvement. The results attained by these two approaches were good.

Recently, the work started by (Cotta and Fernandez, 2004) uses ideas from greedy randomized adaptive search procedures (GRASP) (Feo and Resende, 1995) to develop a hybrid evolutionary algorithm to search for OGRs. A standard evolutionary algorithm is used to search for a set of indexes which in turn are used by GRASP to build a ruler. The results achieved by this approach were good and are being used as a base for the hybridization of GRASP with other techniques for the search of OGRs (Cotta et al., 2006).

Regardless of the technique used, the representation of Golomb Rulers is a decisive aspect on the design and development of the evolutionary algorithms. Next, we describe the most common representations used for this problem, as well as the associated operators and decoding procedures for the indirect representations. They will serve as the basis for our study using landscape analysis.

3.1 Genetic Representations

Finding OGRs is a complex combinatorial optimization problem (Rankin, 1993). Moreover, it has some specific features that distinguish it from other problems with similar characteristics, such as the Travelling Salesperson Problem (TSP). Whilst TSP can be classified as a complete ordered set since the goal is to find a permutation of the n cities that compose the problem, OGRs can be considered an incomplete ordered set (Soliday et al., 1995).

A candidate solution for a particular OGR-n instance can be represented by a sequence of $n-1$ segment's lengths. According to this, an OGR-n is a permutation of $n-1$ elements taken from a set of m elements, where m is defined as the maximum distance between marks (usually $n \ll m$).The construction of such a solution raises several questions: should a maximum value for m be pre-established or should it be adjusted during the construction of a ruler? How to select the $n-1$ elements from a set of m values? How to build a valid permutation with the $n-1$ elements selected?

Another option is to let a candidate solution encode a set of marks to be added to the ruler. With this alternative, given a ruler of length L, each individual contains information about the number of marks that should be inserted and the position where they should be placed.

These two options can be described as finding OGRs by evolving segments, or by evolving marks. The choice of the type of encoding must follow one of these two principles and has a direct influence on the type of algorithm proposed. The set of encodings that we describe follows one of these two principles, although evolution by segments tends to be the most commonly adopted.

3.1.1 Binary Representation

An individual is codified as a binary string with length L, where each bit is associated with an integer position in the ruler. If a given gene has the value 1, it indicates that there is a mark in the corresponding position. The sequence, bounded by an upper length limit, enables the search for rulers without a predetermined number of marks. This implies that binary encoding can be used to evolve both the number and position of marks for a given ruler with a predetermined maximum length L.

Classical crossover operators can be applied. As for mutation, two operators are

used: standard flip and shift mutation. Shift mutation acts in the following way: it randomly selects a mark and shifts it left or right with equal probability. This operator is also used since it is more sensitive to the structure of the solutions.

Additionally, correction and/or insertion procedures can be applied to fix invalid rulers and to see if it is possible to add marks to legal solutions. The line of action is straightforward: a mark is randomly removed if a duplicate measurement is found when checking the segments measured by the ruler. If a valid ruler is obtained at the end of the correction, then the second stage can proceed: an insertion operator tries to add marks in every possible position ensuring that the obtained ruler is still valid. The order in which the positions are selected for insertion is randomly selected. This encoding can be found in (Tavares et al., 2004a) and (Tavares et al., 2004b).

3.1.2 Integer Representation

An integer representation is the simplest and most straightforward encoding for a candidate solution that represents a Golomb ruler through segments length. A chromosome is a vector $v = (v_1, \ldots, v_n)$ where each position v_k belongs to the set $\{1, \ldots, m\}$ for $k \in \{1, \ldots, n\}$ and m is the maximum length for a single segment. This representation allows the duplication of segments which in turn permits unfeasible solutions. A common way to deal with unfeasible solutions relies on the use of penalty-based fitness functions, where the number of duplicate measures and the ruler length act as a penalty. With this encoding, standard genetic operators for crossover and mutation can be used. This encoding has been used in previous studies for comparison purposes since it yields poor results, e.g. (Tavares et al., 2005a).

Additionally, (Feeney, 2003) used a representation which consisted of a set of integers, corresponding to the marks of the ruler. The genetic operators were similar to the ones used by (Soliday et al., 1995): crossover was the same with the addition of a sort procedure; mutation was applied to each ruler segment according to a given probability and consisted in the addition of a random value in the range $[-x, x]$ to the selected segment, where x is the maximum difference between any pairs of marks in any ruler of the initial population.

3.1.3 Permutation Representation

As mentioned before, permutations are typical representations for scheduling and routing problems and have also been applied to the OGR. An n-mark OGR candidate is composed by a permutation of $n-1$ integers, where each one of the integers values represents the length of a segment in the ruler, i.e., $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\}$ denoted by $\pi = (\pi_1, \ldots, \pi_m)$. Decoding it into a feasible solution is done in the following way: the first $n-1$ valid segments from the permutation are used to build the ruler. The segments are selected in such a way that no duplicate measurements exist. This is a deterministic process and segments on the left of the permutation have higher priority, as the ruler is constructed from left to right. Depending on the circumstances, it might happen that in a specific position all segments lead to duplicate measurements. If this situation arises, a random value between 1 and m (with a value high enough to ensure that a valid ruler is always possible to be built) is chosen and a segment with this length is appended to the ruler. During this operation the only restriction that applies is that there should be no duplicated segments. Figure 2 contains the algorithm.

The representation needs genetic operators that can preserve the permutations, such as uniform order based crossover (Gottlieb and Raidl, 2000) or partially-match crossover and swap mutation (Michalewicz, 1992). This encoding has been used as the basis for the representation in (Soliday et al., 1995). The individual contained two

```
1.  Let R be an empty ruler and L a permutation of length m
2.  Let curr_seg = 1
3.  While curr_seg ≤ n − 1 Repeat:
    3.1.  Starting in the beginning of L, find the first segment
    that can be appended to the left of the ruler R built so
    far without leading to duplicate measurements
    3.2.  If such a segment does not exist, randomly select a
    value between 1 and m and append it to the left of R
    3.3.  Increment curr_seg
```

Figure 2: Ruler construction algorithm

lists of segments: one representing the ruler (like a standard integer representation); the second containing a surplus of segments. From these two lists an incomplete permutation was built representing a ruler. Genetic operators were required to guarantee that descendants were also valid permutations to avoid duplicate measurements since Soliday' s representation is an incomplete permutation.

Finally, in some parts of our analysis we will include what we designated by *incomplete permutations*. These can be seen as a mixture between integer and permutation representations, where we have a set of unique integers, e.g., $< 1, 3, 7, 10 >$ but without having all the consecutive integers between the lowest and the highest one. This encoding improves integer representation by not allowing duplicate segments but is less powerful than a complete permutation since it does not have all the possible segments. In our experiments, incomplete representation also uses the same decoder mechanism previously described in figure 2.

3.1.4 Random Key Representation

For the OGR problem, we proposed (Pereira et al., 2003) the encoding of permutations by random-keys, based on the NetKeys extension proposed by (Rothlauf et al., 2002).

Our codification of a solution for an OGR-n follows the same principles as those expressed for NetKeys. Each one of the m positions of the chromosome represents one possible segment. Without loss of generality, we assume that position i corresponds to a segment of length i ($i = 1, \ldots, m$). Also, just like with NetKeys, the value of a given key represents the importance of the related segment. Nevertheless, if we compare both situations there is one additional difficulty associated with OGRs: the interpretation algorithm must not only determine which segments will be part of the ruler but also its relative position. The next step of the process is the same as for standard permutations, resulting in a Golomb ruler. Figure 3 illustrates how decoding and interpretation are related.

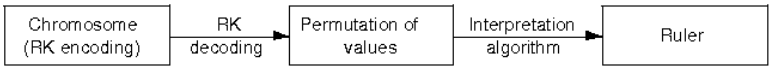

Figure 3: Decoding and interpretation for a random-key individual.

A step-by-step description may help to exemplify how the decoding of the permutation and subsequent interpretation of the information contained in a chromosome is performed. Consider that we are searching for OGR-5 and that the maximum segment

Table 1: Iterations required to construct a Golomb ruler with 5 marks (4 segments) from the permutation {6, 1, 5, 7, 3, 10, 9, 8, 4, 2}

Iteration	Segment Selected	Ruler	Measures
1	6	$\{6, \_, \_, \_\}$	{6}
2	1	$\{6, 1, \_, \_\}$	{1,6,7}
3	3	$\{6, 1, 3, \_\}$	{1,3,4,6,7,10}
4	5	$\{6, 1, 3, 5\}$	{1,3,4,5,6,7,8,9,10,15}

length is 10. Consider also that the chromosome encodes the following key sequence $\{0.87, 0.17, 0.67, 0.27, 0.86, 0.97, 0.71, 0.31, 0.38, 0.40\}$. After performing the decoding, the resulting permutation is $\{6, 1, 5, 7, 3, 10, 9, 8, 4, 2\}$.

During the interpretation phase, the first $n-1$ valid segments from the permutation are used to build the ruler. The iterative algorithm used to build a valid ruler tries to ensure that segments are selected in such a way that no duplicate measurements exist. It is a deterministic process and segments on the left of the permutation have higher priority. The ruler is constructed as shown before in figure 2.

Depending on the circumstances, it might happen that in a specific position all segments lead to duplicate measurements. If this situation arises, a random value between 1 and λ is chosen and a segment with this length is appended to the ruler (see point 2.2 of the algorithm). During this operation the only restriction that applies is that there should be no duplicated segments in the ruler being constructed.

Table 1 illustrates how these steps are applied leading to the construction of the ruler $\{6, 1, 3, 5\}$ from the permutation previously obtained. Note that in the 3^{rd} iteration segments 5 and 7 are not selected because, if inserted in the ruler, they would lead to duplicate measurements. Segment 5 is selected later because its insertion in iteration 4 does not yield any violation.

The genetic operators used for this approach are the standard ones for crossover. In the case of mutation, the operator usually performs positional mutation, i.e, it randomly draws values from the interval $[0, 1]$, according to a distribution, replacing a value at a random position in the chromosome.

3.2 Related Work

The number of works regarding the analysis and study of these techniques is short. To the best of our knowledge, the only analysis research was carried out by (Cotta and Fernández, 2005), where two encodings are considered. One based on the direct representation of solutions, and another on the use of an auxiliary decoder. Some of the properties of the corresponding fitness landscapes are analyzed. The direct encoding showed a highly irregular landscape, whilst the landscape for the indirect representation showed to be regular and exhibiting a fitness-distance correlation comparable to that of the former landscape. The findings were validated in the context of variable neighborhood search.

Further research containing OGRs and evolutionary algorithms can be found in (Cotta et al., 2006). A hybrid evolutionary algorithm with GRASP, tabu-based local search methods and scatter search were also explored. In particular, GRASP and tabu search were embedded into a scatter search algorithm to serve as initialization and restarting methods for the population and as improvement technique respectively. The resulting algorithm is reported to perform well on this problem.

4 Representation Analysis

In the past years, some tools have been used with this aim in mind, from empirical methods to theory-based analysis (e.g., (Manderick et al., 1991a) and (Rothlauf, 2002)). One of these tools is fitness landscape analysis. This type of analysis is concerned with the investigation of the structure of the search space. In a later stage, the characteristics of the search space can be helpful in identifying a search strategy that is likely to improve the algorithm performance. This is an important tool for evolutionary computation.

In spite of much progress that has been made in understanding how evolutionary algorithms work by the use of theoretical investigations, these mathematical analyses are usually limited to simple models. Transposing these analyses to more complex models, especially for hard optimization problems, mathematical analysis techniques seem to be difficult to apply and with limited use. Generalizing the results of empirical analysis is an important step to further understand evolutionary algorithms when applied to combinatorial optimization problems. Lots of work can be found in the literature concerning the analysis of encodings properties, using specific methods e.g. (Rothlauf et al., 2002).

Researchers have also proposed other methods to study representations in general terms, i.e., methods not specific to the algorithms and encodings to the problem at hand. These attempts include the works of (Manderick et al., 1991a), (Sendhoff et al., 1997), (Reeves and Yamada, 1998), (Merz, 2000), (Rothlauf, 2002) and (Raidl and Gottlieb, 2005).

4.1 Fitness Landscapes

The concept of *fitness landscape*, introduced by (Wright, 1932) to demonstrate the dynamics of biological evolutionary optimization, has been useful for the analysis and understanding of evolutionary algorithm's behavior. In addition, the study of fitness landscapes can be of value in designing an evolutionary algorithm since it can help to predict its performance.

Usually, evolutionary search can be represented by three spaces: the *search space*, the *phenotype space* and the *fitness space*. The fitness space reflects the solution quality whilst the search space is made of the candidate solutions. The set of all possible genotypes is denominated *genotype space* and it is equivalent to the search space. This equivalence is possible because variation operators work in the genotype space and an evolutionary algorithm searches for genotypes that decode into phenotypes with high fitness.

Fitness landscapes describe the relation between the *search space* and the *fitness space*. Regarding the *search space* as a landscape, an evolutionary algorithm can be seen as navigating through it in order to find its highest peak. The height of a point in the *search space* (the genotype) reflects the fitness of the decoded solution (the phenotype) associated with that point. The structure of a landscape influences the dynamics of an evolutionary algorithm. Since the representation and operators define the manner how an evolutionary algorithm can perform its search, the choice of a representation and operators for a given problem can be based on the study of the difficulties of the corresponding landscape.

Fitness landscapes is a tuple composed by a set of points (solutions) X, a fitness function f which assigns a numeric value to each solution, and a neighborhood $\mathcal{N}_k$ defined over the set X, given by a distance metric of size k. The following tuple can

represent the fitness landscape:

$$\mathcal{L} = (X, f, \mathcal{N}_k) \tag{1}$$

where:

$$f : X \rightarrow \mathcal{R} \tag{2}$$

and:

$$\mathcal{N}_k(x) = \{y \in X : d(x, y) \leq k\} \tag{3}$$

with a distance metric defined as the minimum number k of applications of an elementary operator m to transform one solution in to another. The fitness landscape can also be viewed as a graph

$$G_{\mathcal{L}} = (X, E) \tag{4}$$

with edge set:

$$E = \{(x, y) \in X \times X \mid d(x, y) = d_{min}\} \tag{5}$$

where d_{min} is the minimum distance between two points in the search space. The diameter of a landscape $diamG_{\mathcal{L}}$, is the maximum distance in the search space.

For binary representations with a bit-flip mutation operator, the solution set is $X = \{0, 1\}^n$, with $G_{\mathcal{L}}$ a hypercube of dimension n and the Hamming distance between bit strings as the distance operator. This yields the minimum distance d_{min} equal to 1 and the maximum distance $diamG_{\mathcal{L}} = n$.

Several methods have been proposed to measure some of these properties (Manderick et al., 1991b), (Jones, 1995), (Reeves, 1999). However, for some of these properties it is not easy to find ways to measure them in a clear and helpful manner. Besides, some of the properties are closely related to each other in many landscapes, making their analysis more difficult to the researcher. Next, we present the most common measures for fitness landscape analysis.

4.2 Fitness Distance Correlation

One way to measure problem difficulty is determining how close is the relation between fitness value and distance to the nearest optimum, in the search space. Fitness distance correlation, called coefficient ϱ, can be estimated by

$$\varrho(f, d) \approx \frac{1}{\sigma_f \sigma_d m} \sum_{i=1}^{m} (f_i - \overline{f})(d_i - \overline{d}) \tag{6}$$

with a given set of points x_i of size m (the random walk length), $f_i = f(x_i)$ the fitness value and $d_i = d_{opt}(x_i)$ the minimum distance to a global optimum solution. The $\overline{f}$ and σ_f are the mean and standard deviation. The search should be easy, for selection-based algorithms, when fitness increases as the distance to the optimum decreases. This indicates the existence of a *path* via solutions with increasing fitness values. A value of -1.0 for ϱ shows that fitness and distance to the optimum are perfectly related, thus the search is easy indicating a strong correlation. A value of $\varrho = 1.0$ indicates the opposite.

4.3 The Autocorrelation Function

The structure of a fitness landscape can be examined by measuring the degree of correlation between points on the landscape. The degree of correlation depends on the difference between the fitness values of the points. Smoother landscapes are highly correlated, making the search for an evolutionary algorithm easier. This is the result of similar fitness values. If the difference of fitness values is higher, the landscape is less

correlated, which implies a rugged landscape, thus being harder to search in it. To measure the ruggedness of a fitness landscape we calculate the *autocorrelation function* $\rho(d)$, which measures the correlation of all pairs of points in the search space with distance d. It can be estimated by computing a large sample of solutions:

$$\rho(d) = \frac{1}{\sigma_f^2 |X_d^2|} \sum_{(x,y) \in X_d^2} (f(x) - \overline{f})(f(y) - \overline{f}) \tag{7}$$

where X_d^2 is the set of all pairs of points in the search space with distance d and $|X_d^2|$ the number of pairs in the set. Another possibility to estimate $\rho(d)$ is to perform a random walk. In this case, the random walk $< f(x_t) >$ defines the correlation of two points s steps away in a m length random walk:

$$\rho_{rw}(s) \approx \frac{1}{\sigma_f^2 (m-s)} \sum_{t=1}^{m-s} (f(x_t) - \overline{f})(f(x_{t+s}) - \overline{f}) \tag{8}$$

If the landscape is *statistically isotropic* (Papoulis and Pillai, 2002), the time series forms a stationary random process, therefore a single random walk is sufficient to obtain ρ_{rw} (Merz, 2000).

4.4 Correlation Length

This measure directly represents the ruggedness of a fitness landscape: the higher the value of the correlation length l, the smoother the landscape is; the lower the value of l, the more rugged the landscape is. The correlation length is defined as

$$l = -\frac{1}{ln(|\rho(1)|)} \tag{9}$$

for $\rho(1) \neq 0$. It is useful to normalize it with the diameter of the landscape:

$$\xi = \frac{l}{diam G_{\mathcal{L}}} \tag{10}$$

The closer the normalized correlation length is to 1, the higher the correlation is. If $\xi = 0$ or close to 0, there is no correlation.

4.5 Other measures

A random walk can also be seen as a time series and, as such, the tools developed to investigate and analyze time series can be applied, to some degree, to fitness landscapes. The analysis of a time series usually involves the identification of a model that represents the generation process of the data. As soon as the model parameters are identified, statistical tests can be applied to see how well the model fits the data. This will allow to see the value of the model for explanation and prediction. The Box-Jenkins approach is a useful statistical method of model building and can be used to measure and express the correlation structure of a fitness landscape. The purpose is to find a model that contains an Auto Regressive and a Moving Average (ARMA). A detailed description of this approach can be found in (Hordijk, 1994, Hordijk and Manderick, 1995).

4.6 Crossover generated Landscapes

Studying the behavior of mutation-based algorithms using the above mentioned methodology is simple and well suited. The reason is that mutation operators are good to generate random walks for correlation analysis. These operators can be seen as producing a time series of fitness values for a succession of solutions (Merz, 2000). Normally, a mutation operator is more appropriate to a certain landscape if the correlation is higher. However, for crossover generated landscapes this is a completely different issue.

According to (Merz, 2000), various attempts have been made to generalize fitness landscape analysis to crossover operators. The first difficulty, when it comes to use crossover as the variation operator is that it works with solutions, the parents, and creates two new solutions (or one, depending on the operator), the offspring. It is then necessary to define how to apply this operator to generate a time series of values. In this work we followed an approach similar to the one defined by (Hordijk and Manderick, 1995). A time series is simply produced by repeatedly applying crossover to two candidate solutions. One of the solutions is always randomly generated whilst the other one starts as a random initial solution but is later replaced by the best of the two offspring. Given the parents and the offspring we measure the fitness and distance to produce the landscape. We have chosen this setup to test crossover with a configuration closer to a more initial stage of an evolutionary algorithm. Populations have not converged and crossover at this stage is more important for the exploration than exploitation of the search space.

With this information the previous fitness distance correlation measure and the correlation are updated for crossover analysis. For both measures, we take into consideration the average fitness of both parents, f_p, the average fitness of both offspring, f_o, as well as the average distance of the children to the known optimum solution, d_{opt}. The measures for analysis are redefined in the following way. Fitness distance correlation for crossover, ϱ_{cx}, can be estimated by

$$\varrho_{cx}(f_o, d_{opt}) \approx \frac{1}{\sigma_{f_o}\sigma_{d_{opt}} m} \sum_{i=1}^{m} (f_{o_i} - \overline{f_o})(d_{o_i} - \overline{d_{opt}}) \tag{11}$$

with a given pair of points x_i of size m (the random walk length). As previously defined for mutation-landscapes, a value of -1.0 for ϱ_{cx} shows that fitness and distance to the optimum are perfectly related, indicating an easy search, whilst a value of $\varrho_{cx} = 1.0$ signals the opposite.

As in (Manderick et al., 1991b), the parent-offspring correlation is given by:

$$\rho_{cx} = \frac{cov(f_p, f_o)}{\sigma_{f_p}\sigma_{f_o}} \tag{12}$$

where $cov(f_p, f_o)$ is the covariance of two variables, in this particular case, the average fitness of the parents and the offspring.

5 Landscape Analysis

In this section, the influence of representation is investigated when looking for optima OGR solutions. This analysis is performed by conducting an investigation on the fitness landscapes defined by different encodings.

We consider four representations: binary, integer, permutation and random-key. For each representation we will study landscapes generated by both variation operators: mutation and crossover. Standard mutation operators are used to create landscapes for each one of the representations. These are: bit-flip and shift mutation, integer flip mutation, swap mutation and uniform flip mutation. For the random-key representation, we will first use the uniform flip operator and later we will switch to a different one. This is accomplished to examine the influence of the operator heuristic bias. As for landscapes generated by crossover, standard one point and uniform crossover operators are applied. The only exception is related to the permutation encoding. In this case, both one point and uniform crossover are altered to allow the manipulation of permutations. Table 2 presents a summary of the encodings and associated operators.

Table 2: Summary of representations and associated operators.

	Binary	Integer	Permutation	Random-key
Mutation	bit-flip shift	integer flip	swap	uniform flip normal flip
Crossover	one point uniform	one point uniform	one point order uniform order	one point uniform

The evaluation of the individuals follow a similar approach to the one described in (Soliday et al., 1995). We consider two criteria: ruler length and feasibility of the solution (i.e., whether it contains repeated measurements). Legal rulers will always have a better fitness value than illegal ones. This has been the principle used on previous research works. In this study we consider the representations in two forms. First, we will be concerned with the simple use of the encodings, then we will add heuristics and/or local optimization techniques. This is accomplished to better examine the effect of heuristics and local methods on the encoding. We begin our analysis with standard and simple encodings that do not use heuristics particularly biased towards fitter phenotypes.

5.1 Experimental Setup

For our analysis, we selected three problem instances: OGR-8, OGR-10 and OGR-12. They are representative of the problem difficulty and can be used, to a certain degree, to test the scalability of the approaches. We performed a comprehensive set of experiments with these instances.

A fitness distance analysis requires that the global optimum is known (or a very near-global optimum solution). For these instances the global optima are known(Soliday et al., 1995), (Pereira et al., 2003). Additionally, the distance between the known optimal solution and the candidate solutions in our random walks is calculated at the phenotype level. This simply means that, when using indirect encodings, all solutions contained in a random walk are converted to its phenotype. The distance is given by the Hamming distance between the solution and the known optimum solution. For the integer-based phenotypes, it is used an adapted Hamming distance formula for integer chromosomes.

For each one of the random walks, we performed 50 runs with 100000 steps. For permutations and random-keys, the maximum segments length used, for OGR-8, OGR-10 and OGR-12, were 50, 75 and 100. For the binary encodings, the maximum individual lengths used are: 60, 80 and 100. For the initialization of the individuals with binary

representation the fill rate of $1s$ is 0.1. These values were chosen to achieve two goals: enable the permutation-based encodings to generate feasible individuals, due to the interpretation process, and to allow the binary representation the capacity to search for rulers with the desired number of marks.

5.2 Mutation Fitness Landscapes

In table 3 we present, for all OGR instances, the results of the applied measures, regarding fitness distance correlation and autocorrelation for mutation landscapes. The first column indicates the representation used, and associated operator when required; BR - binary representation, IR - integer representation, PR - permutation representation, RK - random-key representation and incomplete permutations. The normalized average of the minimum distance between the optimum solution and the best found individual in the random walk can be observed on the next column ($\overline{d_{opt}}$) with the standard deviation in brackets. The last column presents values for the fitness distance correlation (ϱ), followed by the correlation (l) and correlation length (ξ) columns. The values in bold represent the best ones for a given problem instance.

Table 3: Summary results for mutation landscapes.

Representation	Instance	Measures			
		$\overline{d_{opt}}$	ϱ	l	ξ
BR with flip	OGR-8	48.72 (1.35)	-0.75	23.00	0.23
BR with shift	OGR-8	**19.33** (0.39)	-0.84	12.62	0.13
IR	OGR-8	43.20 (0.79)	0.01	1.67	0.15
PR	OGR-8	23.17 (0.77)	**-0.99**	19.77	0.20
RK	OGR-8	22.97 (0.72)	**-0.99**	21.00	0.21
RK with normal	OGR-8	24.33 (0.66)	**-0.99**	**25.00**	**0.25**
Incomplete PR	OGR-8	23.87 (0.28)	-0.46	16.00	0.16
BR with flip	OGR-10	49.17 (1.49)	-0.81	21.00	0.21
BR with shift	OGR-10	**15.17** (0.70)	-0.96	**34.45**	**0.34**
IR	OGR-10	56.60 (1.14)	0.04	1.87	0.17
PR	OGR-10	36.45 (1.06)	**-0.99**	17.56	0.18
RK	OGR-10	36.37 (1.10)	**-0.99**	22.00	0.22
RK with normal	OGR-10	36.89 (0.88)	**-0.99**	25.79	0.26
Incomplete PR	OGR-10	37.25 (0.39)	-0.46	19.85	0.20
BR with flip	OGR-12	49.34 (1.68)	-0.75	23.85	0.24
BR with shift	OGR-12	**12.50** (0.94)	-0.81	**54.60**	**0.55**
IR	OGR-12	65.48 (1.38)	0.07	2.19	0.20
PR	OGR-12	49.86 (1.35)	**-0.99**	21.29	0.21
RK	OGR-12	50.55 (1.21)	**-0.99**	22.00	0.22
RK with normal	OGR-12	49.11 (1.39)	**-0.99**	24.00	0.24
Incomplete PR	OGR-12	49.91 (0.48)	-0.44	20.68	0.21

The data presented in table 3 shows that the normalized average distances between the best found individual during a run and the optimal solution follow the same trend on every OGR instance. As expected, the integer representation shows a larger distance length when comparing to the other encodings. With the single exception of OGR-8, where binary with the flip operator presents worst values than integer representation.

In comparison with all other encodings, and in all instances, the measures values of integer encoding decreases as the complexity of the problem increases. From an average of 43% on the OGR-8, it increases to 56% on OGR-10 and to 65% on OGR-12. The same type of trend can also be seen with permutation and random-key representations. The larger, and complex, the instance is, the larger average distance to the optimum. In spite of this effect, the values are lower than the integer encoding, since this group has an average of 23% for OGR-8, 36% for OGR-10 and 50% for OGR-12. There are no observable differences between permutation and random-keys, as well as with incomplete permutations.

The same is not true for binary representation with flip operator or shift operator. The change of operator shows that binary representation performs differently: with flip mutation the performance is worst (around 49%) than with shift mutation (around 15%). This is an interesting behavior to binary representation since the search space is the same. The reason for this action is the genetic operator used. Flip operator is inserting and deleting marks on the ruler whilst the shift operator just moves a mark a single position. Since individuals are created with few marks, the effect of changing a position of a mark is less severe than introducing or removing new marks. Binary representation is a direct encoding of a Golomb ruler and as such, blind changes made by flip mutation produce more illegal rulers than the more subtle changes made by shift mutation. Later we will see that this is also visible in the fitness distance correlation plots. Another interesting aspect is that binary representation does not decreases its performance when the instance size increases. The performance is more or less stable and in the case of binary representation with shift mutation, there is a slight tendency of improvement. This is a direct result of the number of marks being stable on the individuals and their change of position inside the chromosomes does not induce higher differences of segments size. This is one of the reasons why segments-based representations show higher distances: the length of a single segment can be longer than the others, thus making a ruler distance length higher.

As for incomplete permutations, the first observation is they show better values than integer representation but show similar values to complete permutations (and random-keys). The average distance to the optimum is around 24%, 37% and 50%. These results are more or less expected since the design of this encoding was to use the simplicity of integer representation by using smaller individuals than permutation-based encodings and, at the same time, approximating it more to the structure of Golomb rulers by removing duplicate measurements, which is ensured by the permutation.

In terms of the fitness distance correlation coefficient, the pattern is similar but with some noticeable differences. First, the permutation and random-keys encodings almost achieve the maximum value, regardless of the instance. Second, the performance difference between the mutation operators used with binary representations is smaller. This is an indication that the permutation-based encodings have a good behavior when comparing to the integer and binary representations. With a coefficient value around -0.99, these two encodings (and to a certain extent, mutation operators) seem to be adequate to deal with the problem in question. The behavior of binary representation is dependent on mutation since the coefficient values are not close, although high. As for the integer representation, the fitness distance correlation only confirms the previous findings: values are close to 0 which means no correlation. The last difference concerns incomplete permutations. The encoding attains a behavior inferior to the other permutation-based representations with coefficient values of -0.46 (OGR-8

and OGR-10) and -0.44 (OGR-12).

These observations can be explained. The integer representation performance is a result of its absence of sensitiveness to the problem. In this encoding, a Golomb ruler is a set of unique numerical segments that can be translated into a sequence of numeric marks. For this simple representation, it is possible to have duplicate segments since genetic operators and the methods to create the initial population do not ensure individuals without duplicate segments. Mapping these individuals into legal Golomb rulers will be a difficult task. Nevertheless, even if the genetic operators were modified to prevent duplicate segments, preventing the existence of illegal rulers would still be difficult since duplicate measurements are not verified. Our research just confirms that a simple integer representation is not suitable to solve more complex OGR instances. As for the permutation-based encodings, i.e., permutation and random-keys, the findings are not surprising and can be explained by the decoder associated to these representations. The algorithm that translates a permutation into a Golomb ruler is responsible for the elimination of several illegal rulers since it checks duplicate measurements. The relevant factor is the maximum segment length, which can influence the performance. A small value can allow rulers with duplicate measurements while a larger value can ensure unique measurements. This also helps us to explain why the coefficient values for incomplete permutations are inferior. Since this encoding does not have the total surplus of segments that a complete permutation has, invalid rulers are more easy to be generated making it more difficult to find a path in the landscape to a better solution.

As for binary representation, the interpretation can be given by the adaptation of the mutation operator to the problem. Flip mutation is blind to the problem structure and as such, when inserting or removing a bit, it is in fact inserting or removing a mark in a Golomb ruler. Since there is a high gene epistasis, this type of operation can significantly alter a ruler since the change of a mark in the beginning will affect all the others. The changes with shift mutation are not as drastic as the ones provoked by flip mutation. In this case, shift is more sensitive to the problem structure since it is only moving a mark to a very close position (left or right). Although it can also make significant changes, they will tend to be of lesser effect.

It is also important to observe the fitness distance plots of the several representations. This type of plot presents information about the distribution of the random walk points, in which the fitness is plotted against their minimum distance to an optimum. Figure 4 and 5 contains the fitness distance plots for binary, integer, complete and incomplete permutation and random-key representations, for the OGR-12 instance (which is representative for the analyzed OGR instances). The graphs are not on the same scale since we want to clearly show the pattern given by the representation.

The seven plots help us to confirm and visualize the distribution of the candidate solutions induced by the encodings. From these plots, it is possible to observe the good correlation shown by the permutation-based encodings. These present an ideal distribution of points since the solutions to the optimum improve by *jumping* from one solution to a better one with successively decreasing the jump distance. The same type of conclusion can be drawn to binary representation with some repairs. First of all, the larger concentration of candidate solutions concentrates on the opposite of the optimum solution. In fact, for flip mutation, just a single flow of points converge to the optimum. For shift mutation this is not observable but an ideal distribution is also not found. The line shows a convergence to the optimum but not in a straight way. Integer representation presents a total uncorrelated landscape. Incomplete permutation shows an uncorrelated landscape. In fact, for this encoding, we can observe the landscape mix

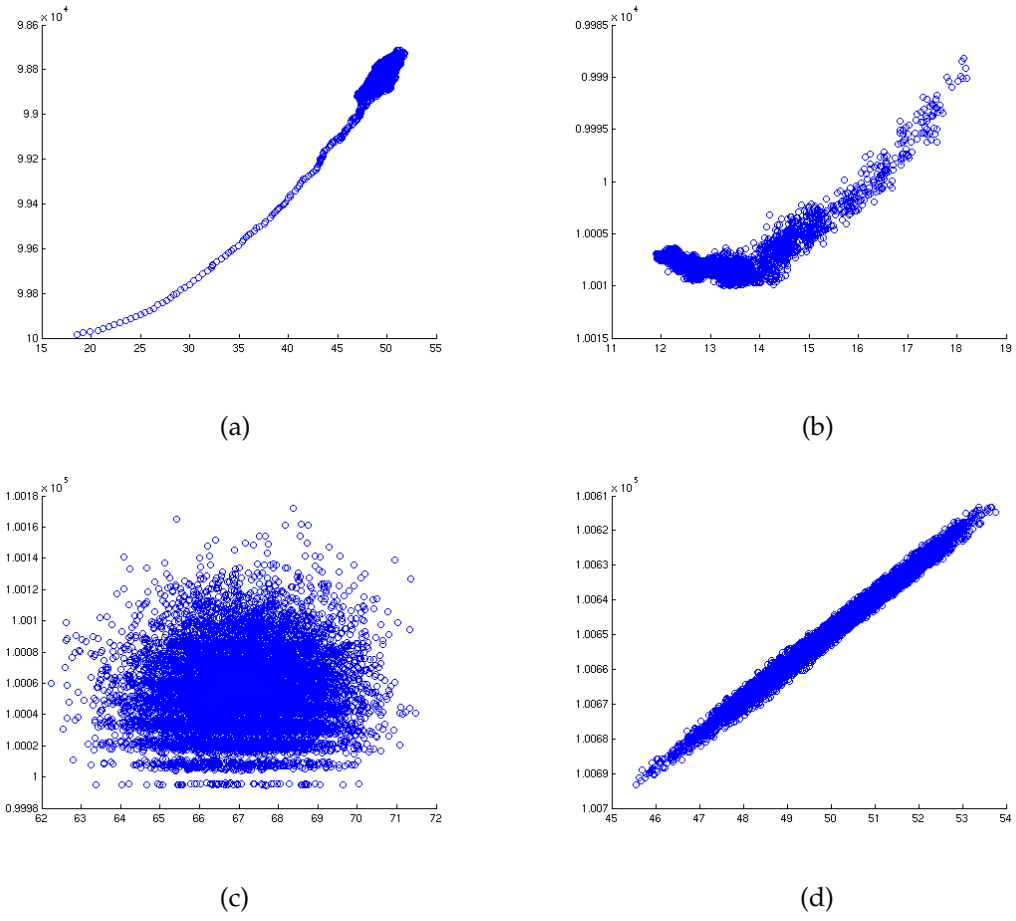

Figure 4: Fitness distance plots for Binary with flip (a), Binary with shift (b), Integer (c) and Permutation (d) for the OGR-12 instance.

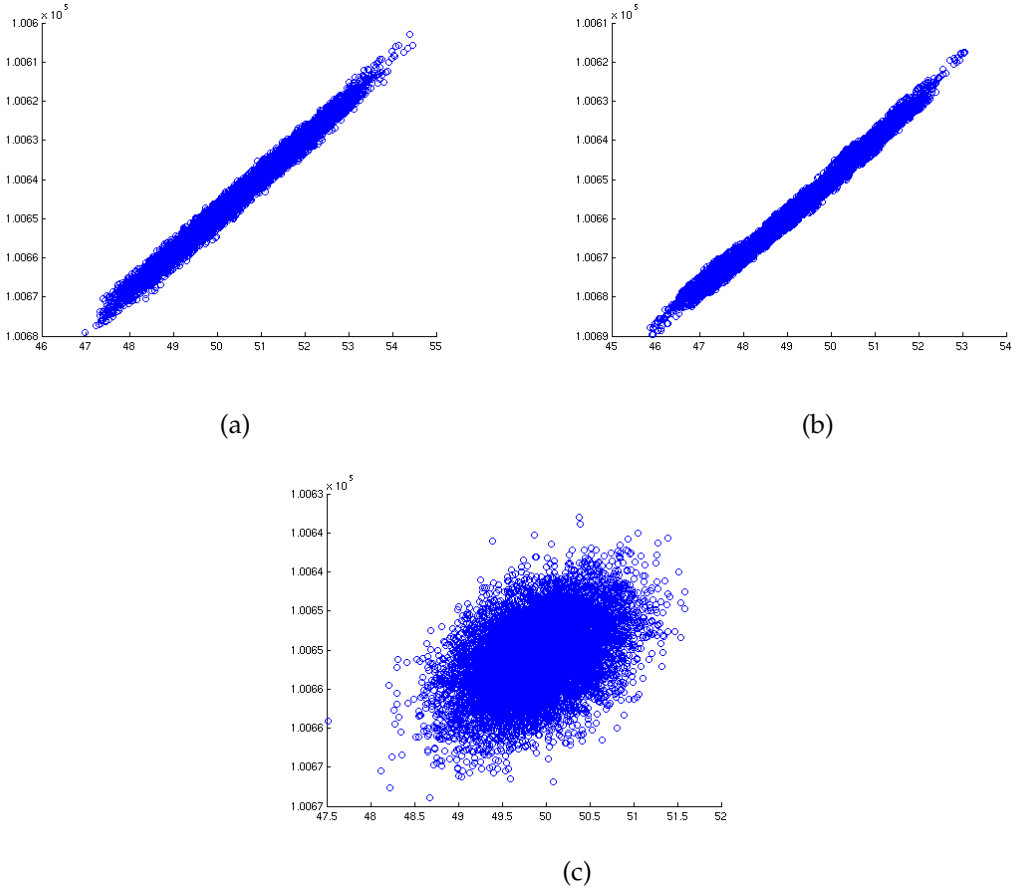

Figure 5: Fitness distance plots for Random key (a), Random key with gaussian operator (b) and Incomplete Permutation (c) for the OGR-12 instance.

of the two encodings that were used as the inspiration for the incomplete permutation. The points distribution follows the same distribution line of permutation-based encodings but with a much larger dispersion on the distribution space, making it similar to the uncorrelated landscape of the integer representation.

In order to gain more discernment, fitness distance correlation should be examined in association with the autocorrelation measures. To determine the correlation length we performed a series of random walks to calculate the autocorrelation function, with a phenotypic distance of 1. The results of the autocorrelation analysis are also presented in table 3. Unfortunately, this coefficient does not provide us valuable information to our analysis. A brief perusal of the results reveals an absence of pattern for all the representations and all the instances. Without significant changes, permutation-based representations have a correlation length, on average, of 0.20 (OGR-8), 0.21 (OGR-10) and 0.22 (OGR-12). Integer encoding present inferior values, but close, to the permutation-based representations. For integer representation the values are 0.15, 0.17 and 0.20. Finally, with binary encoding and flip mutation we report values on the same order, 0.23, 0.21 and 0.24, but not with shift mutation: 0.13, 0.34 and 0.55. Binary representation with shift mutation presents for the last two problem instances values higher than the rest of the encodings.

Overall, all the representations have a similar behavior for autocorrelation. This is an indication that the degree of correlation between points on the landscape produced by these representations is similar. The degree of correlation depends on the difference between the fitness values of the points and the difference of fitness values between points is higher, which implies a rugged landscape, thus being harder to search in it. In fact, this confirms the natural difficulty of the problem. The exception shown by the binary encoding with shift mutation is explained by a higher number of points closer to the optimal solution, as can be seen in the fitness distance correlation plot. This makes a smaller difference of the fitness values between points, turning the landscape a little more smooth.

5.3 Crossover Fitness Landscapes

In this section we will present our analysis on crossover fitness landscapes. We followed an approach similar to the one defined by (Hordijk and Manderick, 1995): a random walk is produced by repeatedly applying crossover to two candidate solutions. One of the solutions is always randomly generated whilst the other one starts as a random initial solution but later is replaced by the best of the two offspring. Given the parents and the offspring we measure the fitness and distance to produce the landscape.

In table 4 we present the results regarding crossover analysis. The table gathers data from one point crossover and uniform crossover. Uniform crossover uses probability $p = 0.5$, for directly transferring the gene of the parent to the offspring (which is commonly used). As before, the first column indicates the representation used, the second column the tested instance, while the last two columns provide the data for the crossover operators. For each operator, we show the normalized average of the minimum distance between the optimum solution and the average offspring found in the random walk ($\overline{d_{cx_{opt}}}$) with the standard deviation in brackets. After that, we present the values for the fitness distance correlation (ϱ_{cx}), followed by the parent offspring correlation (ρ_{cx}).

We include all the representations analyzed before with the sole exception of incomplete permutations. The use of this encoding would require the design of a special

Table 4: Summary results for crossover landscapes.

Representation	Instance	1-Point Crossover			Uniform Crossover		
		$\overline{d_{cx_{opt}}}$	ϱ_{cx}	ρ_{cx}	$\overline{d_{cx_{opt}}}$	ϱ_{cx}	ρ_{cx}
BR	OGR-8	20.96 (0.50)	-0.41	0.83	21.23 (0.51)	-0.40	0.76
IR	OGR-8	23.07 (0.75)	-0.16	0.78	23.27 (0.77)	-0.13	0.74
PR	OGR-8	**14.68** (0.47)	-0.95	**0.97**	19.07 (0.63)	**-0.98**	**0.82**
RK	OGR-8	17.38 (0.55)	**-0.97**	0.83	**17.79** (0.56)	-0.97	0.81
BR	OGR-10	**18.30** (0.39)	-0.48	0.89	**18.48** (0.40)	-0.48	**0.86**
IR	OGR-10	37.31 (1.04)	-0.06	0.81	37.40 (1.04)	-0.04	0.78
PR	OGR-10	24.11 (0.65)	-0.96	**0.97**	30.69 (0.90)	**-0.98**	0.82
RK	OGR-10	28.23 (0.78)	**-0.98**	0.83	28.87 (0.80)	**-0.98**	0.83
BR	OGR-12	**16.18** (0.31)	-0.51	0.90	**16.40** (0.32)	-0.52	**0.85**
IR	OGR-12	52.06 (1.30)	0.00	0.84	52.04 (1.30)	0.02	0.80
PR	OGR-12	34.31 (0.87)	-0.97	**0.97**	43.05 (1.13)	**-0.99**	0.83
RK	OGR-12	39.85 (0.97)	**-0.98**	0.84	40.67 (0.99)	-0.98	0.82

crossover operator, which is outside the scope this research.

A first examination of the results allows us to detect some important aspects. By looking at the best values, marked in bold, it is clear that the representations follow the same trend previously observed with mutation. By looking at the column of the fitness distance correlation coefficient, permutation and random-key outperform all other encodings by a significant margin. In fact, these two representations achieve values close to -1.0 whilst the other encodings are close to 0. The sole exception is binary representation which presents a coefficient value close to -0.5. This pattern is observed on all problem instances and crossover operator.

The observation of the other measures do not present clear patterns as the fitness distance correlation. For the average distance, it is possible to see that permutation-based encodings have an equivalent performance but they do not present the best values. In fact, only the integer representation consistently has the worst values. On the other hand, binary representation presents good distance values. The main reason for this effect is the maximum segment length for the integer and permutation-based encodings. Since this value is not a small one, it is possible to generate a good ruler, closer to the optimum, with a segment fairly large which is not possible with binary representation.

The main question that arises at this point is: why are there not significant differences between the two crossover operators? Looking at the values presented in table 4, it can be stated that these two operators behave in the same way on a problem where position is important. The first justification for this fact is that the operators are not designed to deal with order-based problems. This means that they can operate and attain the same kind of behavior, i.e., equally help to reach good solutions or not.

However, since the epistasis of this problem is high, crossover can have a disruptive effect on the individuals, especially the ones with a direct encoding. Encodings with a decoder and a strong heuristic such as random keys can help to avoid the disrupting effect of these standard operators. Moreover, the same can also be said about

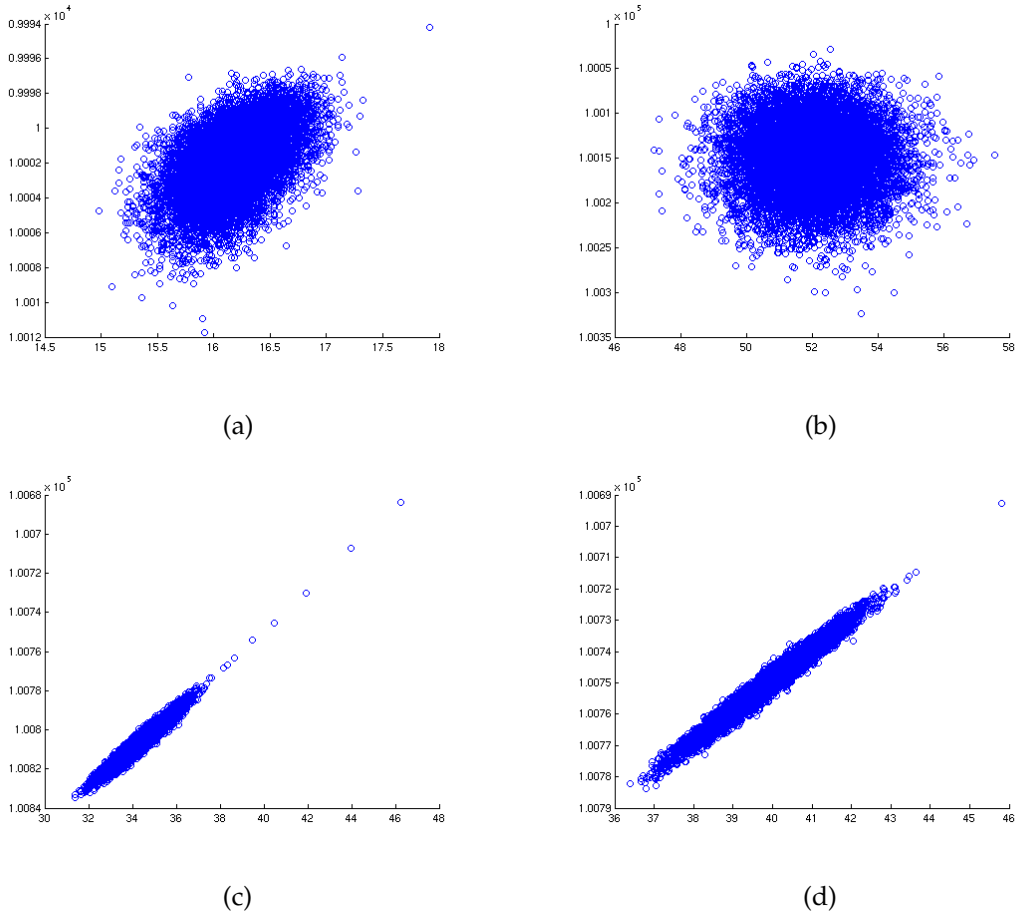

Figure 6: Fitness distance plots for Binary (a), Integer (b), Permutation (c) and Random key (d) for the OGR-12 instance with 1-point crossover.

permutation since the operators are not completely blind to the order and the disruptive effect is also neutralized by the decoding heuristic. In the case of integer permutation and binary representations this does not happen which explains the low values of these measures. For binary encoding, the effect is less severe since the individuals encode marks.

The correlation between parents and their offspring is similar for all representations, attaining high values. As stated before, the specific properties and dynamics of crossover are more difficult to analyze as previous studies have shown (Merz, 2000), as well as the dependency on parental distance (Raidl and Gottlieb, 2005). Since crossover landscapes are generated from pairs of solutions where one of them is always created independently at random, this may ensure a larger parental distance while at the same time it might create a weak dependency on the distance between the parents, allowing a certain loss of homogeneity required to produce fitter offspring. The fitness distance plots for crossover landscapes provide some auxiliary information. Figure 6 presents

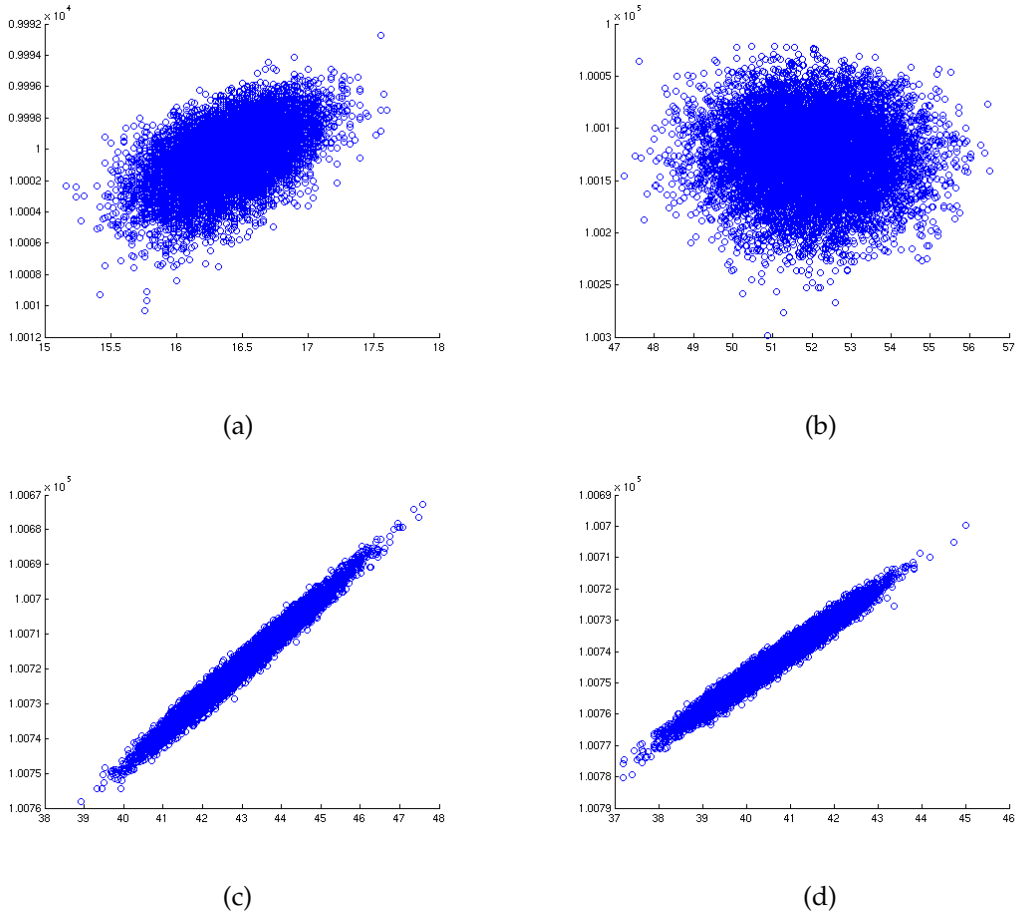

Figure 7: Fitness distance plots for Binary (a), Integer (b), Permutation (c) and Random key (d) for the OGR-12 instance with uniform crossover.

the plots for one point crossover and figure 7 for uniform crossover, on all the tested representations. The first observable fact is that for both operators the shapes of the distribution of the landscapes points are very similar. This is an indication that there are no significant differences between choosing one of the crossover operators. Apart from this, we can find different fitness distance plots for the crossover landscapes on the tested encodings.

Looking at the plots for the binary representation, we can observe an uncorrelated landscape. The distance to the optimum is large as well as between the landscape elements. If the shape of the distribution was more round and closer to a circle, the landscape would be totally uncorrelated. This result is not unexpected since binary representation achieved good coefficient values. The main reason for this fact is that crossover directly manipulates the position of the ruler marks. This change will disrupt the construction of the ruler. As with flip mutation, in a direct encoding a blind operator might cause more damage than good. A total uncorrelated landscape is found, as

expected, with integer representation. In this situation, the distribution form is round and farther away from the optimum. For the remaining representations, the landscapes show that fitness and distance are correlated with similar distribution. In spite of this, it is possible to distinguish a minor difference between the plots of permutation and random-key: the crossover plot for permutation is closer to the optimum. Looking to these fitness distance plots, we can conclude that they are similar to the plots of the mutation landscapes in the case of distribution shape which means that both crossover and mutation are important operators to search for Golomb rulers. This effect can be explained by the stronger heuristic bias in this encoding.

It is possible to conclude for this problem, that the choice of crossover operators, although not very influential on the representation used, play nevertheless an important role. What can be stated is that the choice of the representation is important for crossover to be considered a helpful factor. For permutation-base encodings, crossover is essential for the search process while for other encodings, such as integer and binary representation, these results show that standard genetic operators do not play an important role.

5.4 The Effect of Heuristics and Local Improvement

We will now complement our study by investigating a few simple techniques that can be added to some of the encodings addressed in this research. All decoder-based representations include some simple heuristic mechanisms, essential to its interpretation. Our goal is to perform a study of the additional techniques that were most commonly used in evolutionary algorithms for the search of optimal Golomb rulers.

Some additional tests were made regarding binary, permutation and random-key representations. Our experimentation covers mutations and crossover generated landscapes for these two representations. In our tests, we will consider the following improvements:

- *Binary representation with correction* (BR with C): the standard encoding will be subject to a local method that repairs unfeasible solutions by randomly removing items for a given number of times;
- *Binary representation with insertion* (BR with I): a feasible solution is improved by randomly inserting items on possible positions, as long as it does not generates an unfeasible solution;
- *Binary representation with correction and insertion* (BR with C + I): the union of the two previous steps;
- *Permutation and Random-key with heuristic* (PR / RK with H): the interpretation mechanism gives preference to the smaller segments.

In table 5 we present, for all tested instances, the results of the applied measures for the all types of binary encodings, permutation and two types of random-key using mutation as the variation operator. Table 6 contains the results for the crossover operators. For mutation generated landscapes, it is also included incomplete permutation with heuristic.

The observation of the results generated by mutation operators as described in table 5 reveals one important aspect: permutation-based encodings with heuristic achieved, for all instances, the highest fitness distance correlation value. The result is not surprising since these encodings without the use of heuristic already proved to

Table 5: Summary results for mutation landscapes with heuristics and local improvement techiques.

Representation	Instance	Measures			
		$\overline{d_{opt}}$	ϱ	l	ξ
BR with flip + C	OGR-8	24.05 (0.54)	-0.06	11.50	0.23
BR with flip + I	OGR-8	48.81 (1.37)	-0.79	**13.75**	**0.27**
BR with flip + C +I	OGR-8	24.05 (0.53)	0.00	9.00	0.18
BR with shift + C	OGR-8	19.13 (0.14)	-0.09	11.95	0.24
BR with shift + I	OGR-8	19.66 (0.41)	-0.80	4.14	0.08
BR with shift + C + I	OGR-8	**19.08** (0.15)	-0.39	4.68	0.09
PR with HR	OGR-8	23.14 (0.72)	**-0.99**	10.34	0.21
RK with HR	OGR-8	23.15 (0.74)	**-0.99**	10.00	0.20
RK with normal + HR	OGR-8	22.93 (0.39)	-0.96	8.50	0.17
Incomplete PR with HR	OGR-8	21.99 (0.33)	-0.95	12.26	0.25
BR with flip + C	OGR-10	19.02 (0.34)	-0.08	15.00	0.20
BR with flip + I	OGR-10	49.22 (1.36)	-0.89	13.57	0.18
BR with flip + C +I	OGR-10	19.02 (0.33)	0.00	10.17	0.14
BR with shift + C	OGR-10	14.84 (0.18)	-0.04	10.26	0.14
BR with shift + I	OGR-10	15.36 (0.64)	-0.85	10.99	0.15
BR with shift + C + I	OGR-10	**14.91** (0.17)	-0.53	**18.16**	**0.24**
PR with HR	OGR-10	36.29 (1.01)	**-0.99**	7.91	0.11
RK with HR	OGR-10	36.42 (1.04)	**-0.99**	13.53	0.18
RK with normal + HR	OGR-10	35.80 (0.89)	**-0.99**	11.50	0.15
Incomplete PR with HR	OGR-10	34.27 (0.45)	-0.95	1.67	0.02
BR with flip + C	OGR-12	15.18 (0.24)	-0.23	18.00	0.18
BR with flip + I	OGR-12	49.42 (1.69)	-0.95	21.00	0.21
BR with flip + C +I	OGR-12	15.18 (0.24)	-0.11	26.32	0.26
BR with shift + C	OGR-12	11.81 (0.22)	-0.07	**35.21**	**0.35**
BR with shift + I	OGR-12	12.51 (1.00)	-0.81	14.70	0.15
BR with shift + C + I	OGR-12	**11.65** (0.24)	-0.45	2.92	0.03
PR with HR	OGR-12	49.87 (1.26)	**-0.99**	21.91	0.22
RK with HR	OGR-12	50.28 (1.27)	**-0.99**	20.00	0.20
RK with normal + HR	OGR-12	52.07 (1.26)	**-0.99**	19.01	0.19
Incomplete PR with HR	OGR-12	46.61 (0.54)	-0.96	24.00	0.24

attain a good performance. The unexpected result was the effect of the heuristic in incomplete permutation. This encoding with the heuristic resulted in a large improvement, obtaining coefficient values around -0.95. In terms of the values to the average minimal distance from the best individual to the optimum, we can observe that they are consistent with previous findings.

At a closer examination, we can verify that for binary representations the situation is not easy to analyze. The effects of the different methods are not consistent and to make matters worse, the results seem to indicate that the addition of these mechanisms are not very helpful. Looking to the table, it is clear that these methods produce different and distinct behaviors for this encoding as shown by the reported values. In fact, for most cases, the measures of the encoding decreases considerable. When using correction, fitness distance correlation attains values close to 0, regardless of the operator used.

The exception is when considering the use of correction and insertion with shift mutation. As for insertion, when used alone it seems that it does not have the same effect as correction. In fact, results can be considered good since they are close, on average, to -0.85. What is the reason that can explain these observations? The cause of

this disturbing outcome might be related to the crucial effect of removing marks from a ruler. When correction manipulates the standard encoding it is in fact removing marks from a ruler. When compared with the optimal solution, a legal solution with fewer marks can be more distant than an illegal one since the distance function does not have any kind of information to distinguish these situations.

While an evolution based on encodings with segments has the number of marks fixed, in this case the number of marks is also being evolved. By constantly applying a correction procedure on the binary representation, it could be changing an unfeasible ruler to a feasible one but to a different, smaller, size. With insertion, this *ruler length decreasing effect* is not possible and this fact explains why the performance was not affected. The question still remains for when we join the two methods. In this case, the effect of correction is stronger than insertion. This is corroborated by the fact that insertion alone is not capable of improving the encoding behavior to the same level of permutation-based encodings. It is also true that the comparison might not be entirely fair for all of these situations. Nevertheless, these representations are used for optimization and as such, their results will be compared. Later on we will discuss these two aspects: optimization and analysis.

From the examination of the fitness distance plots in figure 8, it is possible to identify the patterns described above for the binary representation. When correction is involved the distributions are uncorrelated although in a region closer to the optimum. With insertion the distribution plot resembles the plot of the encoding without the local improvement method. This supports the conclusion that the addition of these kind of mechanisms to binary representation for the optimal Golomb ruler problem is not be self evident.

The observation of the plots for the permutation-based encodings in figure 9 shows the expected patterns: a correlated landscape in the same way as the mutation fitness landscapes without heuristic. The relevant fact is the plot for the incomplete permutation: the distribution is considerably different from the one without the heuristic, in spite of being a little more larger than the distributions made by the complete permutation and random-key encodings. The heuristic in this case is very important.

Let's concentrate now in the analysis of crossover generated landscapes. Focusing our attention in the results from table 6, we can see that for crossover generated landscapes the fitness distance correlation, the indirect encodings attain the best results. These results confirm the previous observations. In spite of this fact, it is noticeable that the degree of performance decrease obtained by the use of these extra mechanisms (heuristics and local improvement) is not so wide in comparison with mutation, since they are more or less in the same range. The interesting fact is that insertion has better values with the larger instances while correction decreases them. The combined use of these two mechanisms seem to be ineffective with crossover since the fitness distance correlation coefficient values are close to 0. The type of crossover does not affect the representation's behavior.

In terms of parent-offspring correlation results show that permutation and random-key with heuristic present a steady and constant behavior but binary representation shows a more inconstant pattern. The values range between 0.39 and 0.78 for OGR-8, 0.53 to 0.86 for OGR-10 and from 0.65 to 0.87 for OGR-12, on all the different combinations. This is a result of the direct mapping in which the offspring can change, in terms of fitness, more considerable than the other encodings.

Finally, the fitness distance plots can be examined in figure 10 for binary representation, and in figure 11 for permutation and random-key encodings. The plots show

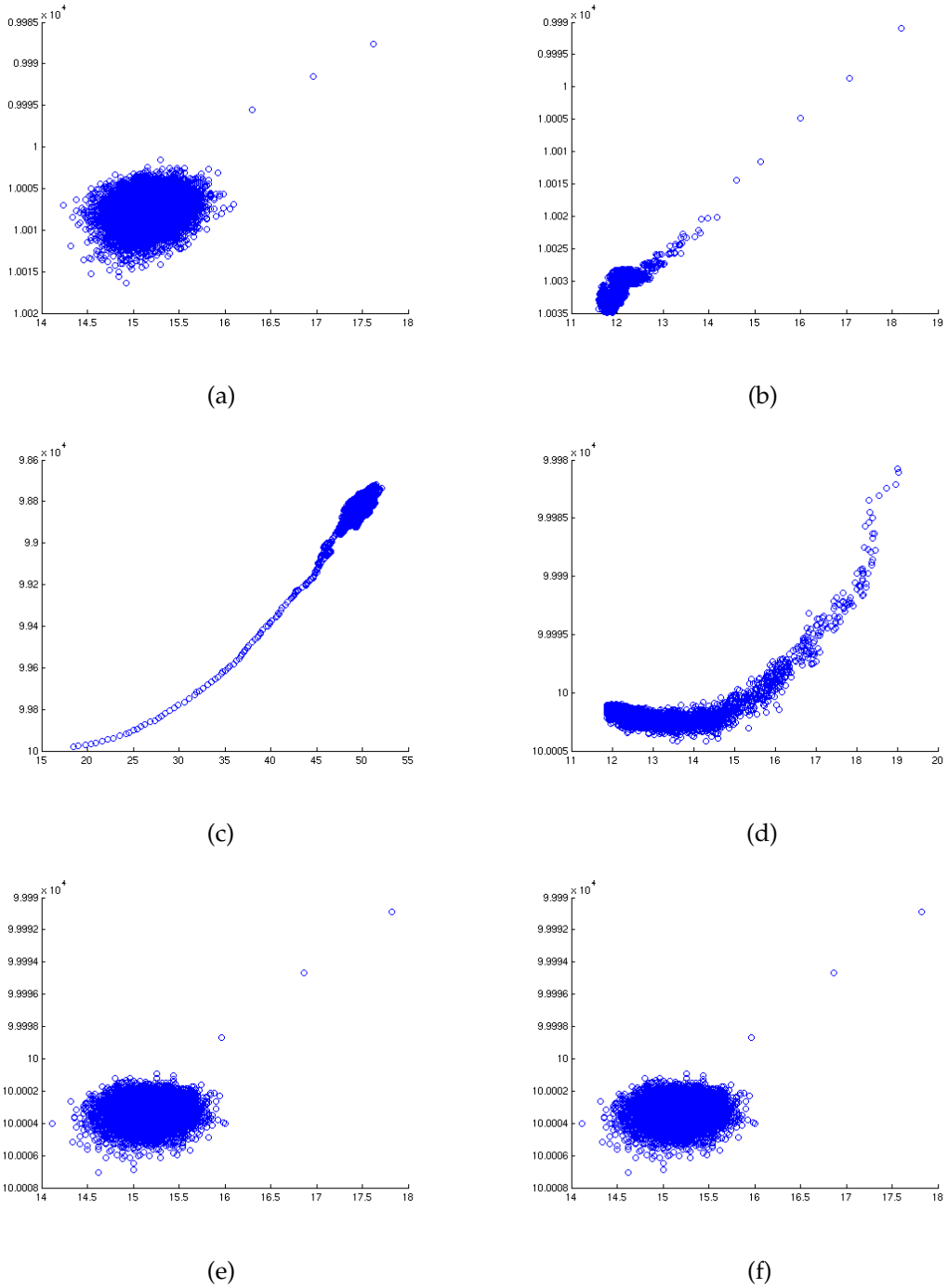

Figure 8: Fitness distance plots for Binary using flip mutation with correction (a), with local improvement (c), with correction and local improvement (e); using shift mutation with correction (b), with local improvement (d), with correction and local improvement (f) on the OGR-12 instance.

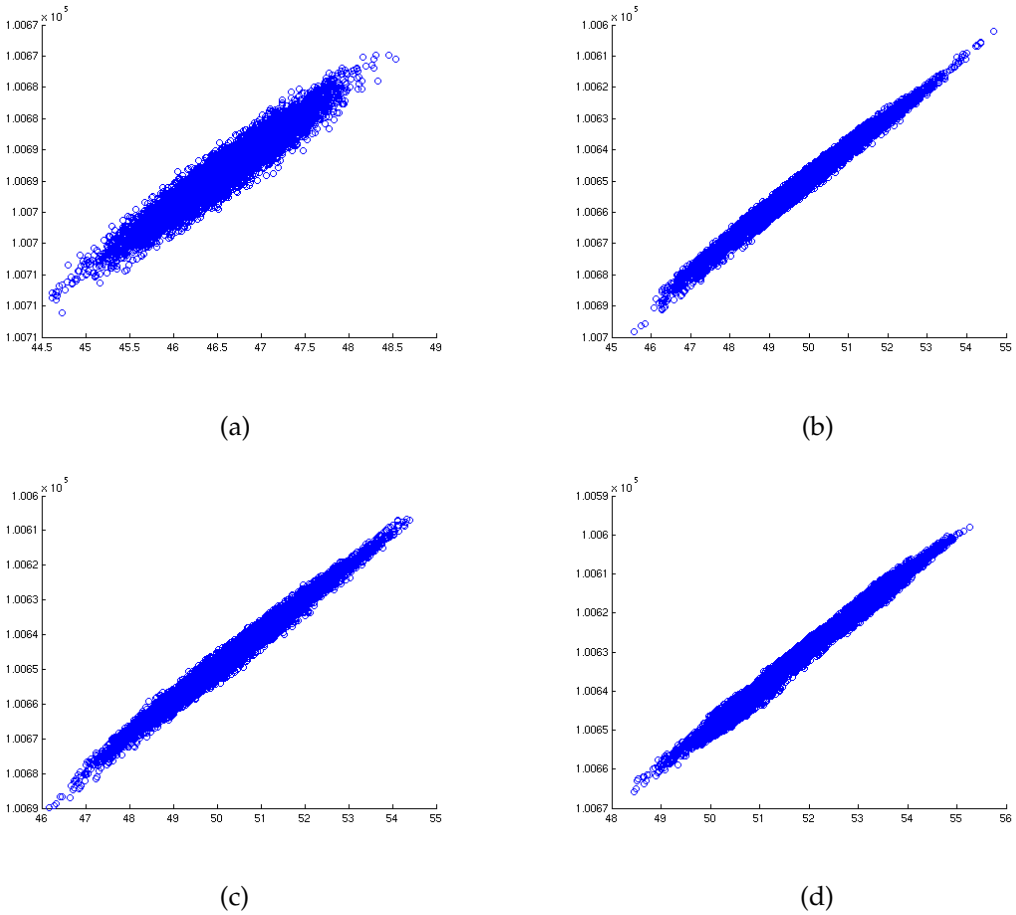

Figure 9: Fitness distance plots for Incomplete Permutation (a), Permutation (b), Random key (c) and Random key with gaussian mutation (d), with heuristic, on the OGR-12 instance.

Table 6: Summary results for crossover landscapes with heuristics and local improvement techniques.

Representation	Instance	1-Point Crossover			Uniform Crossover		
		$\overline{d_{cx_{opt}}}$	ϱ_{cx}	ρ_{cx}	$\overline{d_{cx_{opt}}}$	ϱ_{cx}	ρ_{cx}
BR with C	OGR-8	20.51 (0.47)	-0.44	0.66	20.92 (0.48)	-0.37	0.61
BR with I	OGR-8	21.78 (0.52)	-0.21	0.78	21.71 (0.52)	-0.22	0.70
BR with C + I	OGR-8	21.91 (0.52)	0.02	0.49	21.74 (0.51)	0.02	0.39
PR with HR	OGR-8	**14.70** (0.45)	-0.94	**0.97**	19.05 (0.64)	**-0.98**	**0.82**
RK with HR	OGR-8	17.36 (0.55)	**-0.97**	0.82	**17.80** (0.56)	-0.97	0.81
BR with C	OGR-10	**17.43** (0.34)	-0.30	0.75	**17.58** (0.34)	-0.24	0.68
BR with I	OGR-10	18.47 (0.38)	-0.38	0.86	18.59 (0.40)	-0.40	0.82
BR with C + I	OGR-10	17.94 (0.34)	0.01	0.60	17.88 (0.34)	0.02	0.53
PR with HR	OGR-10	24.10 (0.65)	-0.96	**0.97**	30.70 (0.88)	**-0.98**	**0.82**
RK with HR	OGR-10	28.25 (0.78)	**-0.98**	0.83	28.88 (0.78)	**-0.98**	**0.82**
BR with C	OGR-12	**14.50** (0.24)	-0.25	0.87	**14.59** (0.24)	-0.20	0.81
BR with I	OGR-12	16.18 (0.31)	-0.52	0.86	16.40 (0.32)	-0.55	0.79
BR with C + I	OGR-12	14.59 (0.24)	-0.01	0.75	14.67 (0.24)	-0.03	0.65
PR with HR	OGR-12	34.36 (0.81)	-0.97	**0.97**	43.07 (1.12)	**-0.99**	**0.82**
RK with HR	OGR-12	39.85 (0.99)	**-0.98**	0.84	40.67 (0.98)	-0.98	**0.82**

uncorrelated landscapes for the direct encoding and correlated landscapes for the indirect ones. These results are in agreement with all the previous findings.

This analysis reveals that adding heuristics and/or local improvement methods to a representation is not crucial to obtain better results when solving the optimal Golomb ruler problem. The more important factor is the representation itself and not the auxiliary methods. However, it is important to state that an indirect encoding might have an advantage since the interpretation mechanism already includes a bias toward fitter solutions. Nevertheless, an indirect encoding with operators and mechanism designed in manners not to be blind to the structure of the problem, could present an efficient approach.

5.5 Statistical Analysis

A statistical analysis (Devore and Farnum, 2003, Rice, 2007) is made to confirm the previous findings. The function used for the hypothesis testing is the *Wilcoxon rank sum* test, which is equivalent to the *Mann-Whitney U* test. This test performs a two-sided rank sum test of the hypothesis that two independent samples come from distributions with equal medians. If the null hypothesis is true, the medians are equal. The sets of data are assumed to come from continuous distributions that are identical except possibly for a location shift, but are otherwise arbitrary. The reason to use this test instead of *t-test* is that the former assumes the data to come from a normal distribution with unknown variance. The difference between hypothesis test procedures often arises from differences in the assumptions that the researcher is willing to make about the data sample. In our case, the fitness landscapes generated by the different representations follow different distributions. Although most of them follow normal distributions, e.g., ordinal, permutation and random-key encodings, others simply do not follow it, as confirmed by the *Lilliefors* test. For this reason, it is preferable to use the

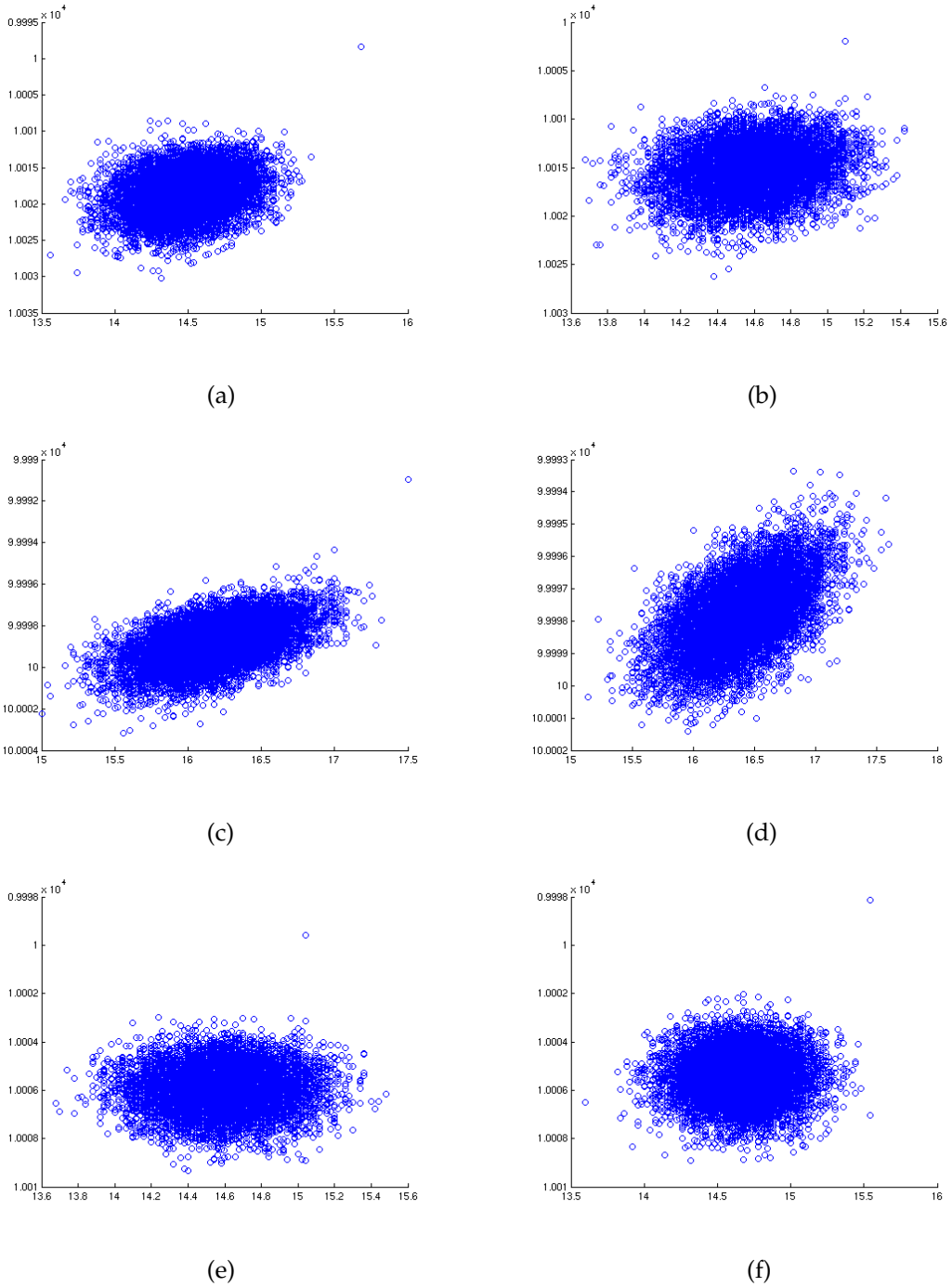

Figure 10: Fitness distance plots for Binary using 1-point crossover with correction (a), with local improvement (c), with correction and local improvement (e); using uniform crossover with correction (b), with local improvement (d), with correction and local improvement (f) on the OGR-12 instance.

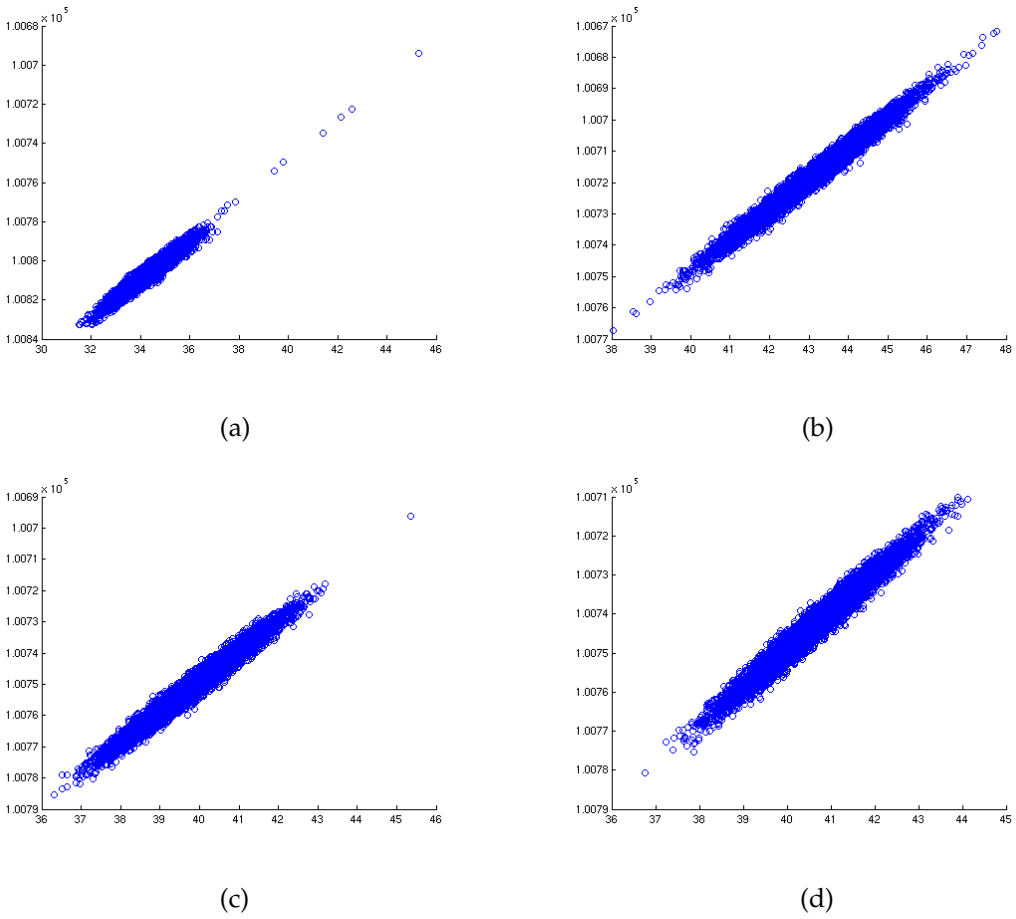

Figure 11: Fitness distance plots for Permutation with heuristic using 1-point crossover (a) and uniform crossover (b), and for Random key with heuristic using 1-point crossover (c) and uniform crossover (d), on the OGR-12 instance.

Table 7: Wilcoxon rank sum test results for mutation landscapes; "+" indicates null hypothesis is true whilst "-" indicates null hypothesis can be rejected at 5% significance level. Lower triangle reports to fitness distance correlation coefficients; upper triangle to distance values.

	BR+flip	BR+shift	IR	PR	RK	RK+normal	iPR
BR+flip	*	+	+	+	+	+	+
BR+shift	+	*	+	+	+	+	+
IR	+	+	*	+	+	+	+
PR	+	+	+	*	-	-	-
RK	+	+	+	-	*	-	-
RK+normal	+	+	+	-	-	*	-
iPR	+	+	+	+	+	+	*

Wilcoxon rank sum test instead of the *t-test*. On all performed tests the significance level is $5\%(\alpha = 0.05)$. We present results for the OGR-12 instance since the other problem instances follow the same trend.

We start our analysis by testing the differences found in the coefficient values contained in all the previous tables. Specifically, it is important to establish what statistical significant differences exist between the distance values, fitness distance correlation values, for mutation and crossover landscapes, as well as the parent-offspring correlation, one point crossover and uniform crossover operators, for all tested representations.

Table 7 shows us the summary for mutation landscapes. The main conclusion is that there are no significant differences between the permutation-based encodings while there are differences between the remaining representations. Even between the binary encoding with the different operator we can find some differences of some statistical significance. This is valid for the distance values and fitness distance correlation values. The table confirms the previous analysis. A similar type of pattern is found when adding the heuristics and local improvement methods. No significance differences are found between the permutation-base encodings, i.e., permutation, random-key with both operators and incomplete permutation. Differences between these encodings and the remaining binary variants are always found. As for binary representation, in terms of distance values, the only significant difference found is between binary with flip mutation and insertion and the other binary variants. In terms of fitness distance correlation coefficient, there are significant differences between the binary variants with insertion when comparing to the other variants, as well as binary with shift operator and correction and insertion.

In terms of crossover, the landscapes follow the same trend of mutation landscapes. The important aspect to consider is: what significant differences can we find between the two genetic operators? Table 8 give us that information.

From the table, we can conclude that there are few cases where we can find significant differences. For fitness distance correlation coefficient, only in two cases we were able to find them: binary with correction and permutation with heuristic. All other encodings reveal no differences between the use of crossover operators. In the case of parent-offspring coefficient, more differences can be found.

It is important to analyze the differences of the coefficient values but also the fitness landscapes. Statistically testing the landscapes can provide us with more insights or

Table 8: Wilcoxon rank sum test results for crossover operators; "+" indicates null hypothesis is true whilst "-" indicates null hypothesis can be rejected at 5% significance level.

	BR	IR	PR	RK	BR+C	BR+I	BR+CI	PR+H	RK+H
ϱ_{cx}	-	-	-	-	+	-	-	+	-
ρ_{cx}	+	-	+	-	-	+	+	+	-

simply confirm the previous findings. In our case, it was the latter option. By analyzing the differences between the landscapes, i.e., analyzing the random walks, we found the same patterns as in the case of just testing the coefficient values test. This was expected since the behavior between the representations are clear and different from each other.

5.6 Optimization Results and Landscape Analysis

The previous analysis provides us some insights in how these different genetic representations can act during an evolutionary process. Despite that, it is important to relate these results with the results attained from optimization results. During the development of our evolutionary approaches to find OGRs, we made several optimization studies. These have been presented and discussed in previous publications: (Pereira et al., 2003), (Tavares et al., 2004a), (Tavares et al., 2004b), (Tavares et al., 2005a) and (Tavares et al., 2005b). Next, we present a brief overview and discussion of the main results, in terms of best rulers found. For a complete description of the optimization results, we refer the readers to the above mentioned publications.

5.6.1 Representation and Best Rulers Found

In table 9 we present the best rulers found for the most established evolutionary approaches for findings OGRs. The table contains the optimization results from instances 5 to 16; the first column indicates the optimal result for the given instance. The columns *Soliday*, *Feeney* and *Cotta* contain the results as described in (Soliday et al., 1995), (Feeney, 2003) and (Cotta and Fernandez, 2004). The following columns contain the results from a standard evolutionary algorithm with an integer representation (colum *EC-Std*), the random-key approach as described in (Pereira et al., 2003) and finally, column *EC-Mark* contains the results from the algorithm with the binary encoding (Tavares et al., 2004a). Bold values indicate where optimum solution have been found.

From this table, we can verify that the evolutionary algorithm with the standard integer representation, *EC-Std*, has the worst performance of all approaches. For the instances of small size, the algorithm is able to find the optimal solutions but as soon as the instances get larger, the performance decreases to values which cannot be considered good. In fact, for the larger and harder instances, the distance is very far (optimum with 177 and the algorithm only reaching 337). This is in agreement with the previous analysis. Looking at the evolutionary approaches of *Soliday* and *Feeney*, we can observe that the results, in spite of being superior the standard approach, they also start to decrease their performance on the harder instances. From OGR-10 they are unable to reach the optimums and for OGR-12, the distance to the optimum is already 21.7% and 18.8%, while for OGR-16 is 34.4% and 20.3%.

These encodings are not proper standard integer representations as defined in our studied. Nevertheless, their implementation is closer to the integer representation than, e.g, o a permutation. The poor results in optimization of these integer-based encodings

Table 9: OGRs lengths for instances between 5 and 16 marks and best results achieved by evolutionary approaches.

Instances	Optimal	Soliday	Feeney	Cotta	EC-Std	EC-Segment	EC-Mark
OGR-5	11	**11**	**11**	**11**	**11**	**11**	**11**
OGR-6	17	**17**	**17**	**17**	**17**	**17**	**17**
OGR-7	25	**25**	**25**	**25**	**25**	**25**	**25**
OGR-8	34	35	**34**	**34**	39	**34**	**34**
OGR-9	44	**44**	47	**44**	61	**44**	**44**
OGR-10	55	62	62	**55**	79	**55**	**55**
OGR-11	72	79	80	74	98	**72**	**72**
OGR-12	85	103	101	94	136	**85**	**85**
OGR-13	106	124	122	111	152	**106**	**106**
OGR-14	127	168	146	135	253	131	**127**
OGR-15	151	206	181	162	294	162	153
OGR-16	177	238	213	189	337	180	**177**

help to support the fact that this type of representations is not suitable for this problem. One of the mains reasons is the lower scalability when compared to other representations/approaches, which is visible from table 9.

The more important columns to look are the last two: *EC-Segment* and *EC-Mark*. From our landscape analysis, permutation-based representations presented to be a good option for this problem. Looking at column *EC-Segment*, it is possible to conclude that this approach is competitive and effective. It finds all the optimal solutions until OGR-13, and for the last three instances the distance to the optimum is very close: 3.1%, 7.2% and 1.7%. This is in agreement with the fitness landscape analysis. Although the optimization results are good, they are not the best.

From the table it is possible to conclude that *EC-Mark*, the evolutionary approach that uses binary representation, presents better results. This evolutionary approach is able to reach all the optimal solutions with the exception of OGR-15 (distance to the optimum of 1.3%). These results can be considered partially unexpected since the fitness landscape analysis showed that, although this encoding can be, at some point, suitable for this problem permutation-based encodings always showed better coefficient results.

This could be considered inconsistent with the previous analysis, however it is important not to forget some factors. Binary encoding was analyzed in different parts, e.g., the mutation operators were studied apart. The optimization results come from an algorithm with several components joined together, as the mutation operators (shift and flip), crossover and the local improvement methods (correction and insertion), as well as the dynamics of the evolutionary process itself. Thus, it is possible that what was considered separately, when combined could, as shown by the optimization results, provide better final results.

Finally, it is important to mention the GRASP approach, as indicated in the third column of the evolutionary approaches. The best rulers found by this approach are better than the initial evolutionary approaches, especially the ones developed with integer representation. Nevertheless, results are inferior to the random-key approach *EC-Segment* and binary *EC-Mark*. GRASP also reveals some problems with scalability:

from OGR-11 to OGR-16 it cannot reach the optimal solutions, although the distances are smaller than the other encodings.

5.6.2 Correction and Insertion

Before concluding this section, it is important to remember that the fitness landscape analysis for the approach with binary representation showed for this encoding peculiar coefficients values when considering the use of correction and insertion. The optimization results of this setting provided some good results and as such, it is necessary to study this issue.

A study on the role of the two procedures, insertion and correction, was previously published in (Tavares et al., 2004b). An overview of the relevant results from that study shows that insertion is essential to obtain good quality Golomb Rulers. The EC algorithm with insertion is capable of reaching optimal solution for rulers with 12 and 13 marks, as well as attaining high quality candidates for the remaining number of marks. In table 10 we present a summary of the best rulers found without insertion. The algorithm was unable to find optimal rulers for any value of L, the maximum number of marks. For larger values of L it could not even obtain rulers with the desired number of marks. Comparing these results with those achieved with insertion, see table 11, we can verify the improvements brought by insertion. For lower values of L the algorithm was able to find the best rulers for instances OGR-12 and OGR-13, while for larger values of L it discovered solutions near the optimal values for instances OGR-15 and OGR-16.

Table 10: Best rulers found without insertion

	C = 0		C = 2		C = 4	
L	n	len	n	len	n	len
100	12	94	12	94	12	100
120	13	116	12	94	12	97
170	14	150	14	149	14	151
200	15	174	15	179	15	181

Table 11: Best rulers found with insertion

	C = 0		C = 2		C = 4	
L	n	len	n	len	n	len
100	12	85	12	85	12	85
120	13	106	13	106	13	106
170	15	155	15	159	15	153
200	16	181	16	186	16	183

In terms of averages of the best solutions found for each of the 30 runs, the results are not very different. Table 12 and table 13 present the averages with and without insertion respectively. Notice that table 12 reports some averages for rulers with lower number of marks. A lower average value does not necessarily means a better performance, since for the same L, rulers with less marks have lower lengths than rulers with

more marks. Once more we can examine the influence of insertion. When using insertion, the distance between the averages and the best solution found ranges between 3.6% and 8.0%. It is important to notice that in this case, almost for all experiments, the algorithm was able to reach the maximum number of marks in 30 runs. When looking at the results without insertion, we can see that there is a higher distance between the averages and the best found solution. These are located between 6.5% and 21.3%, which is clearly worse. Moreover, with the exception of the experiments with $L = 100$ and $L = 120$ with $C = 0$, the number of marks of the best found solution is lower than the maximum number allowed by the set L value.

Table 12: Summary of Averages without Insertion for the best rulers

	C = 0		C = 2		C = 4	
L	avg	std	avg	std	avg	std
100	94.00	n/a	94.00	n/a	100.00	n/a
120	116.00	n/a	114.00	6.18	113.38	5.73
170	161.14	6.49	160.67	6.39	164.16	4.52
200	189.00	8.90	190.67	8.04	192.00	7.39

Table 13: Summary of Averages with Insertion for the best rulers

	C = 0		C = 2		C = 4	
L	avg	std	avg	std	avg	std
100	89.87	2.54	90.17	2.44	89.93	2.32
120	111.13	1.87	111.37	1.94	111.20	1.63
170	163.27	2.92	164.67	2.66	165.27	3.15
200	194.17	3.35	196.44	2.97	195.77	3.30

The number of times the maximum number of marks permitted by the maximum ruler length could be reach, L, is more perceptive in tables 14 and 15. In the first table, without the use of insertion, we can see that only for lower values of L the algorithm can produce rulers with the desired number of marks. Taking into account table 15, there is a huge difference in the performance of the algorithm. For almost every setting, the algorithm was competent in producing Golomb Rulers with the maximum number of marks allowed. When using insertion, the percentage of experiments that produced the maximum number of marks is 100% or close. Without insertion, our approach performs poorly. Even for the best configuration, we were only able to find the maximum number of marks in 50% of the experiments (table 14).

We will concentrate now on the role of correction in the algorithm performance. Examining again tables 14 and 15, it is possible to verify that for different values of C, the results do not change significantly. With the sole exception of the experiment with no insertion and with $C = 4$, the average percentage for the maximum number of marks attained is the same. This apparent exception seems to indicate that with a higher level of correction and no insertion it is hard for the evolutionary algorithm to produce rulers with the maximum number of marks. This is not a surprise since correction implies removing marks and there isn't a procedure to balance it.

Table 14: Percentage of experiments where the maximum number of marks is achieved without Insertion

L	C = 0	C = 2	C = 4
100	33%	50%	8%
120	8%	0%	0%
170	0%	0%	0%
200	0%	0%	0%

Table 15: Percentage of experiments where the maximum number of marks is achieved with Insertion

L	C = 0	C = 2	C = 4
100	100%	100%	100%
120	83%	100%	100%
170	92%	92%	92%
200	83%	100%	83%

From tables 10 and 11 we can concluded that using correction does not influence the best rulers found. These tables show that the results attained are essentially the same. From table 11, we verify that for settings of $L = \{100, 120\}$ the optimal rulers were discovered. For the remaining values of L, we see that the best result for $L = 170$ was obtained with $C = 4$, while for $L = 200$ the best result was attained with no correction. These observations show us that the value of C is not relevant for obtaining good quality solutions. The same behavior is found when insertion is not used. In table 10 we can verify that the use of correction is not able to overcome the lack of insertion. Regardless of the value of C the results attained are equally bad. In terms of averages we can find the same pattern. Tables 12 and 13 show us, independently of using insertion or not, for all levels of correction, the averages are all similar to each other.

Finally, these results show us that correction has a lesser role in the evolutionary process that insertion. This is in agreement with the fitness landscape analysis and helps to explain the evidences found.

6 Conclusion

As for the OGR, it is important to choose a representation that can overcome the difficulties of dealing with duplicate measurements. This is the main reason that encodings with decoders, or that allow the use of heuristics and local improvement methods, can significantly change the performance of an evolutionary algorithm for this problem. The common representations for this problem can encode a solution in two different ways: encoding segments or encoding marks. The option, in terms of representation, can change the behavior of an algorithm significantly.

The fitness landscape analysis for mutation show us that indirect encodings have a good fitness distance correlation and distance to the optimum. Direct encodings is a different issue. The integer representation has the worst behavior in all measures while binary representation presents a better one. Although it does not show the same type of

correlation as the indirect encodings on the fitness distance correlation, the use of shift mutation gives the binary encoding the shortest distance to the optimum. The representations with a stronger heuristic bias are more suitable to solve the problem. For the indirect encodings this is achieved by the decoder whilst for the binary encoding it is attained with genetic operators. Shift mutation is more adequate to the structure of the OGR problem than flip mutation. The same trend can be found on crossover landscapes.

When dealing with the MKP, the use of heuristics and local improvement methods helped to increase the heuristic bias for some of the encodings analyzed and proved to be a good choice. The behavior of the encodings improved with consequences on performance. Regarding the OGR problem and their encodings, the results are less clear and conclusive. First of all, the indirect encodings did not change their behavior. Although a simple heuristic was used, the differences in behavior were minimal or non-existent. Despite that, in terms of performance, some results (absolute solutions) were slightly improved.

For direct encodings, namely binary representation, the results presented in the landscape analysis do not follow a clear pattern. Different behaviors can be found according to the use of an operator with a heuristic and/or local improvement method. In fact, the most evident element is that the use of correction gives indications of not being helpful to the encoding. However, the use of insertion is useful and points to be crucial to binary encoding. These behaviors demonstrate that adding a local improvement method is not always beneficial to an encoding and therefore, to an evolutionary algorithm. These findings are related to optimization results from evolutionary algorithm runs. Permutation-based algorithms are capable of finding good solutions as well as an evolutionary algorithm, with binary encoding and shift mutation using an insertion procedure.

Binary representation can represent a candidate solution more naturally with marks but it needs appropriate local methods and genetic operators, in order to deal with duplicate measurements. On the other hand, permutation-based encodings do not need special genetic operators or local improvement methods to deal with duplicate measurements but they require a good decoding mechanism. The decoder is crucial to their success since it directly influences the heuristic bias.

Acknowledgments

The first author would like to acknowledge grant SFRH/BD/12615/2003 from *Fundação para a Ciência e a Tecnologia* (FCT), Portugal.

References

Bloom, G. and Golomb, S. (1977). Applications of numbered undirected graphs. In *Proceedings of the IEEE*, volume 65, pages 562–570.

Blum, E., Biraud, F., and Ribes, J. (1974). On optimal synthetic linear arrays with applications to radioastronomy. *IEEE Transactions on Antennas and Propagation*, AP-22:108–109.

Chu, P. C. and Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86.

Cotta, C., Dot, I., Fernndez, A., and Hentenryck, P. V. (2006). A memetic approach to golomb rulers. In *Parallel Problem Solving from Nature IX*, pages 252–261. Springer.

Cotta, C. and Fernandez, A. (2004). A hybrid grasp - evolutionary algorithm approach to golomb ruler search. In *Parallel Problem Solving From Nature VIII*, pages 481–489. Springer.

Cotta, C. and Fernández, A. J. (2005). Analyzing fitness landscapes for the optimal golomb ruler problem. In Raidl, G. R. and Gottlieb, J., editors, *Evolutionary Computation in Combinatorial Optimization, 5th European Conference, EvoCOP 2005, Lausanne, Switzerland, March 30 - April 1, 2005, Proceedings*, pages 68–79. Springer.

Devore, J. and Farnum, N. (2003). *Applied Statistics for Engineers and Scientists*. Thomson Brooks/Cole.

Dewdney, A. (1985). Computer recreations. *Scientific American*, pages 16–26.

Dollas, A., Rankin, W., and McCracken, D. (1998). New algorithms for golomb ruler derivation and proof of the 19 mark ruler. *IEEE Transactions on Information Theory*, 44:379–382.

Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer.

Feeney, B. (2003). Determining optimum and near-optimum golomb rulers using genetic algorithms. Master's thesis, Computer Science, University College Cork, Cork, Ireland.

Feo, T. and Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization 6 (1995), 109–133.*, 6:109–133.

Gardner, M. (1972). Mathematical games. *Scientific American*, 1(226):198–112.

Golomb, S. (1972). How to number a graph. In *Graph Theory and Computing*, pages 23–37. Academic Press.

Gottlieb, J. and Raidl, G. R. (2000). Characterizing locality in decoder-based eas for the multidimensional knapsack problem. In *AE '99: Selected Papers from the 4th European Conference on Artificial Evolution*, pages 38–52, London, UK. Springer-Verlag.

Hayes, B. (1998). Collective wisdom. *American Scientist*, 86(2):118–122.

Hordijk, W. (1994). Population flow on fitness landscapes. Master's thesis, Erasmus University Rotterdam, The Netherlands.

Hordijk, W. and Manderick, B. (1995). The usefulness of recombination. In *Third European Conference on Artificial Life*, pages 908–919. Springer-Verlag.

Jones, T. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, New Mexico.

Klove, T. (1989). Bounds and construction for difference triangle sets. *IEEE Transactions on Information Theory*, IT-35(879–886).

Manderick, B., de Weger, M., and Spiessens, P. (1991a). The genetic algorithm and the structure of the fitness landscape. In *4th International Conference on Genetic Algorithms*, pages 143–150.

Manderick, B., de Weger, M., and Spiessens, P. (1991b). The genetic algorithm and the structure of the fitness landscape. In *4th International Conference on Genetic Algorithms*, pages 143–150. Morgan Kaufmann.

Merz, P. (2000). *Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Eective Search Strategies*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, third edition.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization, Algorithms and Complexity*. Dover Publications, Mineola, New York.

Papoulis, A. and Pillai, S. U. (2002). *Probability, Random Variables and Stochastic Processes*. McGraw-Hill.

Pereira, F. B. (2002). *Estudo das Interaces entre Evoluo e Aprendizagem em Ambientes de Computao Evolucionria*. PhD thesis, University of Coimbra.

Pereira, F. B., Tavares, J., and Costa, E. (2003). Golomb rulers: The advantage of evolution. In *Proceedings of the 11th Portuguese Conference on Artificial Intelligence*, Beja, Portugal. Springer-Verlag.

Raidl, G. R. and Gottlieb, J. (2005). Empirical analysis of locality heriability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. *Evolutionary Computation Journal*, 13(4):441–475.

Rankin, W. T. (1993). Optimal golomb rulers: An exhaustive parallel search implementation. Master's thesis, Duke University.

Reeves, C. R. (1999). Fitness landscapes and evolutionary algorithms. In *AE '99: Selected Papers from the 4th European Conference on Artificial Evolution*, pages 3–20, London, UK. Springer-Verlag.

Reeves, C. R. and Yamada, T. (1998). Genetic algorithms, path relinking and the flowshop sequencing problem. *Evolutionary Computation Journal*, 6(1):230–254.

Rice, J. A. (2007). *Mathematical Statistics and Data Analysis*. Thomson Brooks/Cole.

Rothlauf, F. (2002). *Representations for Genetic and Evolutionary Algorithms*. Springer.

Rothlauf, F., Goldberg, D., and Heinzl, A. D. (2002). Network random keys - a tree representa-tion scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10(1):75–97.

Sendhoff, B., Kreutz, M., and Seelen, W. V. (1997). A condition for the genotype-phenotype mapping: Casualty. In *7th International Conference on Genetic Algorithms*, pages 73–80.

Shearer, J. (1990). Some new optimum golomb rulers. *IEEE Transactions on Information Theory*, IT-36:183–184.

Soliday, S., Homaifar, A., and G., L. (1995). Genetic algorithm approach to the search for golomb rulers. In *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pages 528–535. Morgan Kaufmann.

Tavares, J., Leito, T., Pereira, F. B., and Costa, E. (2005a). Evolving segments length in golomb rulers. In *7th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA05)*, Coimbra, Portugal. Springer-Verlag.

Tavares, J., Pereira, F. B., and Costa, E. (2004a). Evolving golomb rulers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Seattle, USA. Springer-Verlag.

Tavares, J., Pereira, F. B., and Costa, E. (2004b). Understanding the role of insertion and correction in the evolution of golomb rulers. In *Proceedings of the Congress of Evolutionary Computation*, Portland, USA. IEEE Press.

Tavares, J., Pereira, F. B., and Costa, E. (2005b). Golomb rulers: Experiments with marks representation. In *7th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA05)*, Coimbra, Portugal. Springer-Verlag.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proceedings of the VI International Conference on Genetics, Vol. 1*, pages 356–366.