Golomb Rulers: A Fitness Landscape Analysis

Jorge Tavares, *Member, IEEE*, Francisco B. Pereira, *Member, IEEE*, and Ernesto Costa

***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 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.**

I. Introduction

One of the fundamental questions in the development of evolutionary algorithms is the representation of candidate solutions for a given problem [1]. 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.

The study of representations and operators in evolutionary algorithms is a recurrent topic in the area of the evolutionary computation [2], [3]. 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. These include the works of [2], [4], [5], [6], [3] and [7].

J. Tavares is with the French National Institute for Research in Computer Science and Control (INRIA), Research Center Lille-Nord Europe, 59650 Villeneuve d'Ascq, France (email: jorge.tavares@ieee.org).

F. B. Pereira is with CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal, and also with the Department of Computer Science, Instituto Superior de Engenharia de Coimbra (ISEC), Polytechnic Institute of Coimbra, 3030 Coimbra, Portugal (e-mail: xico@dei.uc.pt).

E. Costa is with CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal (e-mail: ernesto@dei.uc.pt).

Golomb Rulers are numerical sequences, or a class of undirected graphs, named after the relevant work of the mathematician Solomon Golomb [8], [9], that find application in a variety of engineering fields [10]. A good example is in the field of communications. Other examples of areas where they are used include X-ray crystallography, or error detection and correction in the field of coding theory.

In this work we present a contribution towards a better understanding of the role of representation when evolutionary algorithms are applied to the Optimal Golomb Ruler problem. This paper is structured as follows: a description and formal definition of the problem is presented in section II; section III presents an overview of evolutionary techniques applied to Golomb rulers; in section IV we describe the analysis techniques used for this work; experimental results and discussion are reported in section V; finally, in section VI we present some conclusions.

II. Problem Description

Golomb Rulers are numerical sequences, or a class of undirected graphs, named after the relevant work of the mathematician Solomon Golomb [8], [9], that find application in a variety of engineering fields [10]. A good example is in the field of communications. When setting up an interferometer for radio astronomy, placing the antennas on the marks of a Golomb ruler maximizes the recovery of information about the phases of the signal received [11], [12]. Other examples of areas where they are used include X-ray crystallography [9], or error detection and correction in the field of coding theory [13], [14].

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 [13]. 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.

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

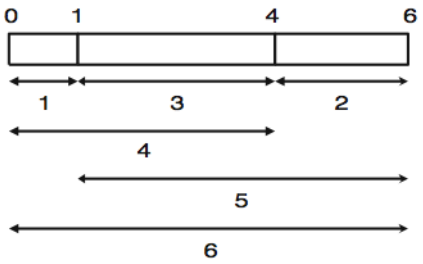


Fig. 1. A Perfect Golomb Ruler with 4 marks.

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. 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).

III. Evolutionary Approaches

The first work regarding the application of evolutionary algorithms to Golomb Rulers was the approach proposed by Soliday [15]. This work consisted of a standard evolutionary algorithm with simple modifications to allow the search for OGRs. It adopted a representation where an OGR candidate is a permutation of segments, built from two lists containing segments of the ruler. The results achieved were not very good. Feeney [16] proposed and studied a hybrid approach to search for Golomb rulers. His work was a continuation of [15] and consisted in applying local improvement techniques with evolutionary algorithms. Achieved results were an improvement to the ones attained by [15].

Pereira and Tavares developed two different approaches for finding OGRs. The first approach [17] uses 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 [18]. That way standard genetic operators could be used. The second approach [19] followed a different strategy. It uses binary representation to represent a candidate solution. In this particular case, the candidate solution represents the position of marks in a Golomb Ruler instead of the segments length. The results attained by these two approaches were good.

The work started by [20] uses ideas from greedy randomized adaptive search procedures (GRASP) [21] 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 [22].

The number of works regarding the analysis for the OGR is short. To the best of our knowledge, the only work was carried out by Cotta [23], 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. Further research containing OGRs and evolutionary algorithms can be found in [22].

IV. Representation Analysis and Fitness Landscapes

In the past years, some tools have been used for representation analysis, from empirical methods to theory-based analysis (e.g., [2] and [3]). One of these tools is fitness landscape analysis introduced by [24]. 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. Researchers have also proposed other methods to study representations in general terms, and these include the works of [2], [4], [5], [6], [3] and [7].

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.

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 [25], [26], called coefficient ϱ, can be estimated by

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

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.

V. Landscape Analysis on the OGR

We consider four representations: binary (as in the work of [19]), integer, permutation and random-key (as in the work of [17]). For each representation we will study landscapes generated by standard mutation operators. These are: bit-flip and shift mutation, integer flip mutation, swap mutation and uniform flip mutation. The evaluation of the individuals follow a similar approach to the one described in [15]. 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. 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 [15], [17]. 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.

A. Simple Encodings

In table I we present, for all OGR instances, the results of the applied measures. The first column indicates the representation used, and associated operator when required; BR - binary representation [27], IR - integer representation [28], PR - permutation representation [29] and RK - random-key representation [30]. 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).

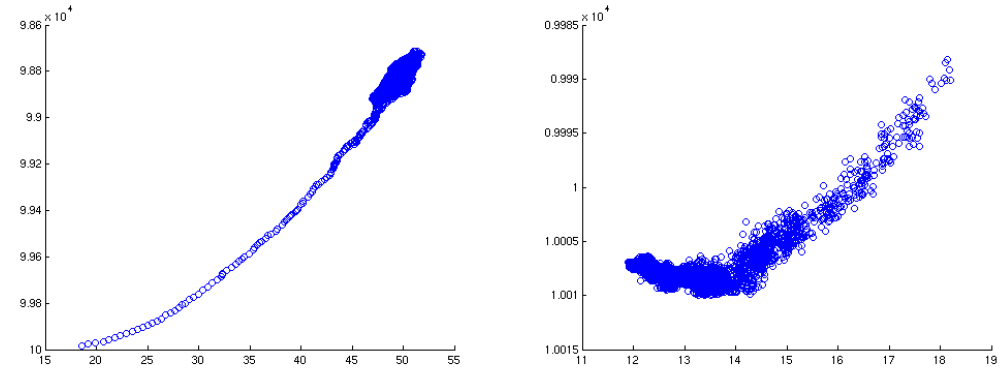
The data presented in table I shows the same trend on every OGR instance. As expected, the integer representation shows a larger distance length when compared to the other encodings. With the single exception of OGR-8, where binary with the flip operator presents worst values than integer representation. In comparison with all other encodings, and in all instances, the measures values of integer encoding decrease 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.

TABLE I
Summary results for mutation landscapes.

Representation	Instance	$\overline{d_{opt}}$	ϱ
BR with flip	OGR-8	48.72 (1.35)	-0.75
BR with shift	OGR-8	**19.33** (0.39)	-0.84
IR	OGR-8	43.20 (0.79)	0.01
PR	OGR-8	23.17 (0.77)	**-0.99**
RK	OGR-8	22.97 (0.72)	**-0.99**
BR with flip	OGR-10	49.17 (1.49)	-0.81
BR with shift	OGR-10	**15.17** (0.70)	-0.96
IR	OGR-10	56.60 (1.14)	0.04
PR	OGR-10	36.45 (1.06)	**-0.99**
RK	OGR-10	36.37 (1.10)	**-0.99**
BR with flip	OGR-12	49.34 (1.68)	-0.75
BR with shift	OGR-12	**12.50** (0.94)	-0.81
IR	OGR-12	65.48 (1.38)	0.07
PR	OGR-12	49.86 (1.35)	**-0.99**
RK	OGR-12	50.55 (1.21)	**-0.99**

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. Another interesting aspect is that binary representation does not decrease 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.

In terms of the fitness distance correlation coefficient, the



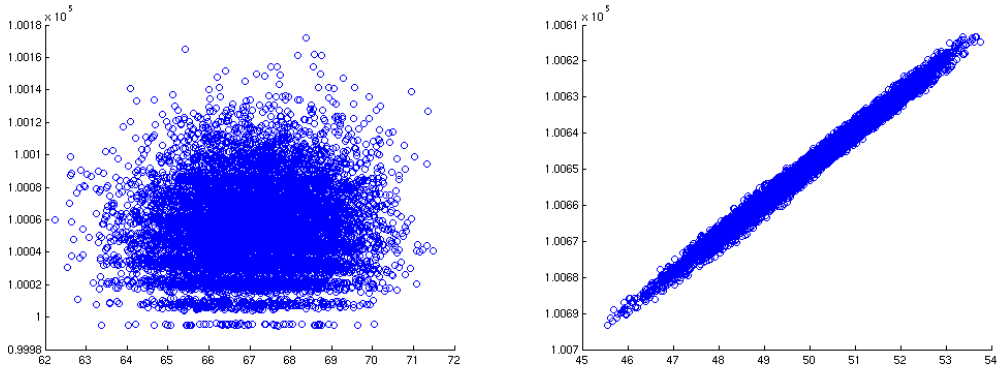


Fig. 2. Fitness distance plots for Binary with flip (a), Binary with shift (b), for Integer (c) and Permutation (d) on the OGR-12 instance.

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 compared 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.

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.

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 (figure 2). 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. 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.

B. 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 paper. 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 regarding binary representation. 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.

In table II we present, for all tested instances, the results of the applied measures for the all types of binary encodings using mutation as the variation operator. 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 decrease considerably. 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.

TABLE II
RESULTS WITH HEURISTICS AND LOCAL IMPROVEMENT TECHNIQUES.

Representation	Instance	Measures	
		$\overline{d_{opt}}$	ϱ
BR with flip + C	OGR-8	24.05 (0.54)	-0.06
BR with flip + I	OGR-8	48.81 (1.37)	-0.79
BR with flip + C +I	OGR-8	24.05 (0.53)	0.00
BR with shift + C	OGR-8	19.13 (0.14)	-0.09
BR with shift + I	OGR-8	19.66 (0.41)	**-0.80**
BR with shift + C + I	OGR-8	**19.08** (0.15)	-0.39
BR with flip + C	OGR-10	19.02 (0.34)	-0.08
BR with flip + I	OGR-10	49.22 (1.36)	**-0.89**
BR with flip + C +I	OGR-10	19.02 (0.33)	0.00
BR with shift + C	OGR-10	14.84 (0.18)	-0.04
BR with shift + I	OGR-10	15.36 (0.64)	-0.85
BR with shift + C + I	OGR-10	**14.91** (0.17)	-0.53
BR with flip + C	OGR-12	15.18 (0.24)	-0.23
BR with flip + I	OGR-12	49.42 (1.69)	**-0.95**
BR with flip + C +I	OGR-12	15.18 (0.24)	-0.11
BR with shift + C	OGR-12	11.81 (0.22)	-0.07
BR with shift + I	OGR-12	12.51 (1.00)	-0.81
BR with shift + C + I	OGR-12	**11.65** (0.24)	-0.45

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.

C. 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 findings with the results attained from optimization runs. In table III we present the best rulers found for the most established evolutionary approaches for findings OGRs. The table contains the optimization results from instances 8 to 16; the first column indicates the optimal result for the given instance. The columns *Soliday* and *Feeney* contain the

TABLE III

OGRs LENGTHS FOR INSTANCES BETWEEN 8 AND 16 MARKS AND BEST RESULTS ACHIEVED BY EVOLUTIONARY APPROACHES.

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

results as described in [15] and [16]. 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 [17] and finally, column *EC-Mark* contains the results from the algorithm with the binary encoding [19]. Bold values indicate where optima solutions 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. The algorithm is not able to find the optimal solutions and as soon as the instances get larger, the performance decreases to values which cannot be considered good. Looking at the evolutionary approaches of *Soliday* and *Feeney*, we can observe that the results, in spite of being superior to 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%. Observing 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. 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. 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. 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 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.

VI. CONCLUSION

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 fitness landscape analysis for mutation show us that indirect encodings have a good fitness distance correlation and distance to the optimum. Direct encodings are a different issue. 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.

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.

REFERENCES

[1] K. Deb, "Introduction to representations," in *Evolutionary Computation 1: Basic Algorithms and Operators*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Institute of Physics Publishing and Oxford University Press, 2000, ch. 14, pp. 127–131.

[2] B. Manderick, M. de Weger, and P. Spiessens, "The genetic algorithm and the structure of the fitness landscape," in *4th International Conference on Genetic Algorithms*, 1991, pp. 143–150.

[3] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*. Springer, 2002.

[4] B. Sendhoff, M. Kreutz, and W. V. Seelen, "A condition for the genotype-phenotype mapping: Casualty," in *7th International Conference on Genetic Algorithms*, 1997, pp. 73–80.

[5] C. R. Reeves and T. Yamada, "Genetic algorithms, path relinking and the flowshop sequencing problem," *Evolutionary Computation Journal*, vol. 6, no. 1, pp. 230–254, 1998.

[6] P. Merz, "Memetic algorithms for combinatorial optimization problems: Fitness landscapes and efective search strategies," Ph.D. dissertation, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2000. [Online]. Available: citeseer.ist.psu.edu/article/merz01memetic.html

[7] G. R. Raidl and J. Gottlieb, "Empirical analysis of locality heriability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem," *Evolutionary Computation Journal*, vol. 13, no. 4, pp. 441–475, December 2005.

[8] S. Golomb, "How to number a graph," in *Graph Theory and Computing*. Academic Press, 1972, pp. 23–37.

[9] G. Bloom and S. Golomb, "Applications of numbered undirected graphs," in *Proceedings of the IEEE*, vol. 65, 1977, pp. 562–570.

[10] W. T. Rankin, "Optimal golomb rulers: An exhaustive parallel search implementation," Master's thesis, Duke University, 1993.

[11] E. Blum, F. Biraud, and J. Ribes, "On optimal synthetic linear arrays with applications to radioastronomy," *IEEE Transactions on Antennas and Propagation*, vol. AP-22, pp. 108–109, 1974.

[12] B. Hayes, "Collective wisdom," *American Scientist*, vol. 86, no. 2, pp. 118–122, 1998.

[13] A. Dollas, W. Rankin, and D. McCracken, "New algorithms for golomb ruler derivation and proof of the 19 mark ruler," *IEEE Transactions on Information Theory*, vol. 44, pp. 379–382, 1998.

[14] T. Klove, "Bounds and construction for difference triangle sets," *IEEE Transactions on Information Theory*, vol. IT-35, no. 879–886, 1989.

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

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

[17] F. B. Pereira, J. Tavares, and E. Costa, "Golomb rulers: The advantage of evolution," in *Proceedings of the 11th Portuguese Conference on Artificial Intelligence*. Beja, Portugal: Springer-Verlag, December 2003.

[18] F. Rothlauf, D. Goldberg, and A. D. Heinzl, "Network random keys - a tree representa-tion scheme for genetic and evolutionary algorithms," *Evolutionary Computation*, vol. 10, no. 1, pp. 75–97, 2002.

[19] J. Tavares, F. B. Pereira, and E. Costa, "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, June 2004.

[20] C. Cotta and A. Fernandez, "A hybrid grasp - evolutionary algorithm approach to golomb ruler search," in *Parallel Problem Solving From Nature VIII*. Springer, 2004, pp. 481–489.

[21] T. Feo and M. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization 6 (1995), 109–133.*, vol. 6, pp. 109–133, 1995. [Online]. Available: citeseer.ist.psu.edu/resende02greedy.html

[22] C. Cotta, I. Dot, A. Fernndez, and P. V. Hentenryck, "A memetic approach to golomb rulers," in *Parallel Problem Solving from Nature IX*. Springer, 2006, pp. 252–261.

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

[24] S. Wright, "The roles of mutation, inbreeding, crossbreeding and selection in evolution," in *Proceedings of the VI International Conference on Genetics, Vol. 1*, 1932, pp. 356–366.

[25] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, L. Eshelman, Ed. San Francisco, CA: Morgan Kaufmann, 1995, pp. 184–192. [Online]. Available: citeseer.ist.psu.edu/jones95fitness.html

[26] T. Jones, "Evolutionary algorithms, fitness landscapes and search," Ph.D. dissertation, University of New Mexico, Albuquerque, New Mexico, May 1995.

[27] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[28] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer, 2003.

[29] D. Whitley, "Permutations," in *Evolutionary Computation 1: Basic Algorithms and Operators*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds. Institute of Physics Publishing and Oxford University Press, 2000, ch. 17, pp. 139–150.

[30] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, 1994.