Evolutionary Golomb Rulers

Jorge Tavares[1], Francisco B. Pereira[1,2], and Ernesto Costa[1]

[1] ECOS - Evolutionary and Complex Systems Group
Centre for Informatics and Systems of the University of Coimbra,
Polo II - Pinhal de Marrocos, 3030 Coimbra, Portugal
[2] Instituto Superior de Engenharia de Coimbra,
Quinta da Nora, 3030 Coimbra, Portugal
{jast, xico, ernesto}@dei.uc.pt

Abstract. In this paper we present a new evolutionary algorithm designed to efficiently search for optimal Golomb rulers. The proposed approach uses a redundant random keys representation to codify the information contained in a chromosome and relies on a simple interpretation algorithm to obtain feasible solutions. Experimental results show that this method is successful in quickly identifying good solutions and that can be considered as a realistic alternative to massive parallel approaches that need several months or years to discover high quality Golomb rulers.

1 Introduction

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. They were named after the relevant work of the mathematician Solomon Golomb [1], [2] and, unlike usual rulers, they have the ability to measure more discrete measures than the number of marks they carry. Also Golomb rulers are not redundant, since they do not measure the same distance twice.

Fig. 1. A perfect Golomb ruler with 4 marks

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 [3]. There may exist multiple different OGRs for a specific number of marks. A perfect Golomb ruler is a particular case of an OGR: 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. As you can see, 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 [3].

OGRs are used in a wide range of real world situations. For example, 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 [4], [5]. Other examples of areas where they are used include X-ray crystallography [2] or error detection and correction in the field of coding theory [3], [6].

For a small number of marks it is possible to construct OGRs by hand. As the number of marks increases, the problem becomes 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 massively parallel brute force algorithms.

In this paper we present a new Evolutionary Computation (EC) approach designed to efficiently search for good solutions. Our goal is to introduce a robust technique that has the ability to quickly identify good solutions (if possible optimum solutions) and that can be considered an alternative to brute force methods, that usually need too much time to obtain an answer and so cannot be considered as a realistic option in real world situations. The proposed algorithm uses random keys to codify possible solutions, a technique which has already showed to be effective in situations where the chromosome needs to encode a permutation [7]. We will also analyze how the addition of a simple heuristic to the EC algorithm may further improve its search performance.

Results presented here show that our evolutionary approach is successful in quickly identifying high quality rulers with several marks, confirming that they are a well-balanced technique capable of discovering good solutions within a reasonable time.

The structure of the paper is the following: in section 2 we provide a formal definition of Golomb rulers. Then, in section 3 we present an overview of some of the main techniques used to generate and verify OGRs. Section 4 comprises a description of the proposed EC model. In section 5 we present and analyze the most important experimental results achieved, while in section 6 we examine if the addition of a simple heuristic to the EC algorithm may further improve its search performance. Finally, in section 7 we draw some overall conclusions and suggest direction for future work.

2 Golomb Rulers

In this section we present a formal definition of Golomb rulers. A 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 on position 0, whereas the length of the ruler is given by the position of the rightmost mark an. 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. This way, 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 exists no other n-mark ruler having a smaller largest mark a_n. In such a case a_n is called the length of the n-mark OGR (OGR-n, for short).

Finding OGRs is a complex combinatorial optimization problem. Moreover, it has some specific features that differentiate it from other problems with similar characteristics, such as the Travelling Salesperson Problem (TSP). Whilst TSP can be classified as a complete ordered set (the goal is to find a permutation of the n cities that compose the problem), OGR can be considered as an incomplete ordered set [8]. Assume that we represent a ruler by a sequence composed by its segment's lengths. The 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 poses several difficulties:

- 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?

When we present our approach in section 4, we will describe how we addressed each one of these problems.

3 Related Work

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.

One of the most important classes of algorithms proposed, aimed to construct OGRs from the scratch. Dewdney presented one of these methods in 1985 [9]. The proposed technique is composed of two phases: the ruler generation and the Golomb 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 [10] and Dollas et. al. [3] 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 [11] proposed a depth first backtracking algorithm. During 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 by adjoined to the ruler without leading to an invalid solution. The algorithm backtracks when too few integers remain eligible to allow 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[3] and distributed.net[4]. 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 OGR-20, OGR-21, OGR-22 and OGR-23. Search for OGR-24 and OGR-25 started in July 2000.

To our knowledge there was just one attempt to apply EC techniques to search for OGRs. In 1995, Soliday et al. proposed a genetic algorithm designed to find good solutions for different Golomb ruler instances [8]. They adopted a representation where a n-mark OGR candidate is composed by a permutation of $n-1$ integers, each one of these values representing the length of a segment in 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. Also special genetic operators were required to guarantee that descendants were also valid permutations. Evaluation of individuals considered two fitness criteria: ruler length and number of repeated measures. Results presented in the above-mentioned paper are not very good. Best solutions evolved for rulers with 10 to 16 marks are far from the known overall bests (the lengths of the best rulers found are between 10% and 36% higher than the optimum values).

In table 1 we present the length of OGRs up to 23 marks. Also presented is the length of the best solutions found by the evolutionary approach proposed by Soliday et. al.

4 An Evolutionary Approach with Random Keys

The representation chosen for individuals plays a crucial role on the performance of EC algorithms. In the problem addressed in this paper, a candidate solution for a particular OGR-n instance must specify, either the position of the n marks, or the length of each one of the $n-1$ segments. We adopted the second approach, so a chromosome is composed by a permutation of integers representing successive

[3] GVANT project. Available at http://members.aol.com/golomb20/

[4] Distributed.net "Project OGR". Available at http://www.distributed.net/ogr/

Table 1. OGRs lengths for instances between 5 and 23 marks and best results achieved by Soliday's EC approach

Instances	Best Solutions	Soliday's EC approach
OGR-5	11	11
OGR-6	17	17
OGR-7	25	25
OGR-8	34	35
OGR-9	44	44
OGR-10	55	62
OGR-11	72	79
OGR-12	85	103
OGR-13	106	124
OGR-14	127	168
OGR-15	151	206
OGR-16	177	238
OGR-17	199	-
OGR-18	216	-
OGR-19	246	-
OGR-20	283	-
OGR-21	333	-
OGR-22	356	-
OGR-23	372	-

segment lengths. The maximum segment length λ is specified as a parameter of the evolutionary algorithm and remains unchanged during search.

4.1 Chromosome Representation and Interpretation

Even though λ is known, there are two crucial decisions to make when building a solution for OGR-n: how to select $n-1$ distinct segments from the set of λ values and how to build a valid permutation with the selected elements. We adopted a representation that tries to deal efficiently with this situation:

- It provides a straightforward way to select which elements will compose the permutation;
- It finds a natural arrangement for the selected segments.

Also, the representation chosen for individuals enables a simple application of standard genetic operators guaranteeing that most of the generated descendants are legal solutions.

In our approach, the chromosome is composed by a permutation of λ distinct values. Encoding of the permutation is done with random keys (RK). RK were introduced by Bean [7] in 1994 and obtained good results in situations where the relative order of the tasks is important [7], [12]. One of their main advantages is that it is possible to apply standard genetic operators to chromosomes (e.g.,

one point or uniform crossover) and still obtain feasible individuals. We will just present a brief overview of RK. For a detailed description, consult [7] or [13]. RK representation uses a sequence of N random numbers to encode a permutation of length N. These numbers are typically sampled from the real interval $[0, 1]$. Both the position and the value of the keys are important for the interpretation of the sequence. To obtain the permutation that corresponds to a given key sequence, all keys are sorted according to their values in decreasing order. Then, the original positions of the keys will be used to construct the permutation. For example, consider the following key sequence $r = \{0.5, 0.7, 0.3, 0.9, 0.4\}$. Position 4 contains the highest value of the key sequence (0.9), so 4 will be the first element of the resulting permutation. Then, the next highest value is at position 2. The ordering process continues in a similar way and at the end we get the permutation $\{4, 2, 1, 5, 3\}$. From a key sequence of length N we can always construct a permutation of N unique numbers between 1 and N (or between 0 and $N - 1$ if needed).

In [13], Rothlauf et. al. proposed NetKeys, an extension of RK to problems dealing with tree network design. The situation addressed is that of the design of a minimum spanning tree over a fully connected graph with n nodes. In these circumstances, a NetKey sequence will be composed by $L = \frac{n(n-1)}{2}$ random numbers (the number of links in the graph). Positions are labelled and each one represents one possible link in the tree. The value of a particular key can be interpreted as the importance of the link it represents. The higher its value, the higher the probability that this link is used in the construction of the tree. From this sequence, a permutation of L numbers can be constructed in the same way as described for standard RK. Then the construction of the tree begins: links are added to the tree in an order that is in accordance to the value of its key. If the insertion of a link would create a cycle, then it is skipped and construction continues with the next one. The process comes to an end when $n-1$ links have been selected.

Our codification of a solution for an OGR-n follows the same principles as those expressed for NetKeys. Each one of the λ positions of the chromosome represents one possible segment. Without loss of generality, we assume that position i corresponds to a segment of length i ($i = 1, \ldots, \lambda$). Also, just like with NetKeys, the value of a given key represents the importance of the related segment. If we compare both situations, there is nevertheless one additional difficulty associated with OGRs: the interpretation algorithm must determine, not only which segments will be part of the ruler, but also its relative position. Figure 2 illustrates how decoding and interpretation (the two stages required to assign fitness to an individual) are related.

Fig. 2. Decoding and interpretation of the information contained a chromosome

A step-by-step description helps 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 λ 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 RK 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 from left to right, as shown in figure 3.

1. Let R be an empty ruler and L a permutation of length λ constructed from the key sequence encoded in the chromosome. Let $curr\_seg = 1$.
2. **While** $curr\_seg \leq n - 1$ **Repeat:**
  2.1. 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
  2.2. If such a segment does not exist, randomly select a value between 1 and λ and append it to the left of R.
  2.3. Increment $curr\_seg$

Fig. 3. Ruler construction algorithm

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 R.

Table 2 illustrates how these steps are applied leading to the construction of the ruler $\{6, 1, 3, 5\}$ from the permutation previously obtained.

Table 2. 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}

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.

4.2 Evaluation

To evaluate an individual we follow a similar approach to the one described in [8]. We consider two criteria: ruler length and legality of the solution (i.e., whether it contains repeated measurements). The equation used to assign fitness to an individual x is the following:

$$fitness(x) = \begin{cases} \text{K - number of repeated measurements} & \text{, if } x \text{ is illegal} \\ \text{K + M - Ruler length} & \text{, if } x \text{ is legal} \end{cases} \quad (1)$$

Where K is a large constant (we used the value 1000 in our experiments) and M is an upper bound for the ruler length (for an OGR-n instance we set $M = n \times \lambda$). This formula ensures that:

- The fitness of a legal solution is always higher than the fitness of an illegal solution;
- Invalid solutions are ranked according to their illegality (number of repeated measurements);
- Legal solutions are ranked according to their length. Shorter rulers receive higher fitness.

5 Experimental Results

To evaluate our approach (which henceforth we will designate by RK-EC) we performed a set of experiments with several OGR instances. More precisely, we used the evolutionary algorithm to seek for good rulers with 10 to 19 marks.

The settings of the EC algorithm are the following: Number of generations: 20000; Population size: 200; Tournament selection with tourney size: 5; Elitist strategy; $\lambda = 30$ for instances with number of marks ≤ 13, $\lambda = 50$, for instances with 14 and 15 marks and $\lambda = 60$ for instances with number of marks ≥ 16; Two-point crossover with rate: 0.75; An evolutionary strategy like mutation operator is used. When undergoing mutation, the new value v_{new} for a given gene (i.e. a key in the chromosome) is obtained from the original value v_{old} in the following way:

$$v_{new} = v_{old} + \sigma \times N(0, 1) \quad (2)$$

Where $N(0, 1)$ represents a random value sampled from a standard normal distribution and σ is a parameter from the algorithm. In our experiments we used $\sigma = 0.1$. Mutation rate was set to 0.25 per gene.

For every OGR instance we performed 30 runs with the same initial conditions and with different random seeds. All initial populations were randomly

generated with values for keys selected from the real interval $[0, 1]$. Significance of the results was tested with a t-test with level of significance 0.01.

Values for different parameters were set heuristically. Nevertheless, and even though we did not perform an extensive parametric study, we conducted some additional tests and verified that, within a moderate range, there was not an important difference in the outcomes. In table 3 we summarize, for all instances, the results achieved by the RK-EC approach. Column Best shows the length of the best solution found, whereas column Average presents the averages of the best solutions found in each one of the 30 runs. To allow an easy comparison, column Best solutions shows the length of the different OGRs.

Table 3. Results obtained by the RK-EC algorithm for all instances

Instances	Best Solutions	RK-EC	
		Best	Average
OGR-10	55	55	58.9
OGR-11	72	72	74.6
OGR-12	85	91	93.8
OGR-13	106	111	115.1
OGR-14	127	134	138.6
OGR-15	151	160	166.2
OGR-16	177	193	197.8
OGR-17	199	221	233.6
OGR-18	216	266	274
OGR-19	246	299	319

A brief perusal of the results reveals that our approach was able to consistently find good quality solutions. It found the optimal rulers for instances with 10 and 11 marks. For OGR-n instances with $12 \leq n \leq 17$ it found rulers whose length is not more than 10% larger than the optimum solutions. When applied to instances OGR-18 and OGR-19 the RK-EC approach had more difficulties and the lengths of the best rulers found are approximately 20% bigger than the OGRs. This is not an unexpected result. Golomb rulers define search spaces whose topology is very hard to sample. One reason contributing to the difficulty in finding good solutions is the number of interactions that occur between the genes that compose a chromosome. There is a high epistasis associated with this problem, since changing the position of a mark (or the length of a single segment) affects all other marks and a large number of measurements carried out by a given ruler. As the number of marks increases, it is likely that there is a growth in the difficulty of controlling the effects of changes in the ruler.

Despite these difficulties, for all instances where the comparison is possible (OGR-10 to OGR-16), the RK-EC algorithm found rulers clearly shorter than those ones discovered by the previous EC approach (consult table 1 for comparison). These results show that an EC approach can be a reliable option in

situations where good quality rulers have to be quickly discovered and there is not enough time to apply massive parallel techniques that take months or even years to produce any outcome. Even in the worst cases, a 20% departure from the optimum might be considered satisfactory if the answer is found in a few minutes instead of 1 year. Also, as it can be seen from table 3, the averages of the best results found in the 30 runs are not very distant from the best solution found (this distance never exceeds 7%) showing that the RK-EC algorithm is a robust approach.

In our experiments, we selected a fairly large value for λ. It varied between 30 (for instances with no more than 13 marks) and 60 (for instances with more than 15 marks). It is known that the length of the biggest segment for OGR-10 is 13, whilst for OGR-19 is 40. As for the other instances, values are somewhere between these extremes. The gap that exists between these values and the chosen λ is likely to affect the performance of the EC algorithm. On one hand, by setting λ to a large value helps to obtain valid individuals in the interpretation phase, which might benefit the search process. On the other hand, as the chromosomes encode a large number of segments they will also contain a lot of redundant information. As an example, when seeking for OGR-19, from the 60 segments belonging to the chromosome only 18 are used to construct a ruler. This situation might slow down (or even prevent) convergence to areas of the space where good solutions are. Results presented in this paper are inconclusive in what concerns this situation and further experiments are needed to determine the real influence of λ in the performance of the evolutionary algorithm. One possible alternative that we will analyze in a near future is to let the value of λ evolve during the simulation.

6 Addition of a Simple Heuristic

In this section we will describe a final set of experiments that will enable us to analyze if the addition of a heuristic may further improve the performance of the proposed approach. In the past few years, there has been numerous attempts to combine EC algorithms with other methods, such as problem specific heuristics or generic local search techniques. In many situations, results achieved give a clear indication that this hybridization is advantageous since it enhances the ability of the evolutionary algorithm to discover good solutions [14], [15], [16].

In this paper we will just describe a straightforward test performed with a basic heuristic that favors the existence of small length segments in a ruler. Even though the occurrence of some small segments does not ensure that the fitness of a ruler is high (depending on the situation, it might be preferable to have two medium sized segments than one small and one big), it is clear that typical good solutions for OGRs will tend to contain elements with these characteristics.

In accordance to this assumption, we added to the interpretation algorithm, a heuristic that favors the insertion of small segments. This way, the modified algorithm that uses the information decoded from the chromosome to construct a ruler has the following steps presented in figure 4.

```
1. Let R be an empty ruler, let L be a permutation of
length λ constructed from the key sequence encoded
in the chromosome and let O be an ordered sequence
of λ integers {1, 2, ..., λ − 1, λ}. Let curr_seg = 1.
2. While curr_seg ≤ n − 1 Repeat:
   2.1. R ← UniformRandomValue(0, 1)
   2.2. If R ≥ φ then
        2.2.1. Selected permutation P ← L
   2.3. Else
        2.3.1. Selected permutation P ← O
    2.4. Starting in he beginning of P, find the first
   segment that can be appended to the left of the
   ruler R built so far without leading to
   duplicate measurements
   2.5. If such a segment does not exist, randomly
   select a value between 1 and λ and
   append it to the left of R.
   2.6. Increment curr_seg
```

Fig. 4. Interpretation algorithm

The relevant difference from the first version of the algorithm is that, with a probability ϕ, the next segment to be inserted in the ruler is selected from an ordered sequence of λ integers instead of the permutation decoded from the chromosome. Although the modification in the interpretation algorithm is minimal, when the heuristic is applied, it will introduce some variation in the construction of the ruler (the sequence order is different from the permutation which was decoded from the chromosome). Moreover this variation is not random, since it favors small segments. Anyway, this heuristic in no way guarantees that good solutions are obtained. When selected, it will always try to insert segments following the same order. Given this deterministic behavior, it might even induce premature convergence to local optima. That's why the application of this heuristic will be done at a moderate rate.

To examine if this modification influences the performance of the search algorithm we repeated all previous experiments. We used the same experimental settings and set $\phi = 0.25$. Results are presented in table 4.

Even though variations in the results attained are not dramatic, there is nevertheless evidence that the addition of the described heuristic improves the search performance of the algorithm. In this new set of experiments, the optimum was found for all in-stances with $n \leq 13$. Moreover, with the exception of OGR-15 and OGR-19, best rulers found by RK-EC with heuristic are shorter than the ones discovered by RK-EC alone.

Another result that supports the advantage of the hybrid algorithm is that the average length of the best solutions found in each of the 30 runs is always smaller in experiments that used the heuristic. With the exception of OGR-15, OGR-16 and OGR-19, these differences are statistically significant.

Table 4. Results obtained by the RK-EC algorithm with heuristic for all instances

Instances	Best Solutions	RK-EC		RK-EC Heuristic	
		Best	Average	Best	Average
OGR-10	55	55	58.9	55	55.0
OGR-11	72	72	74.6	72	73.5
OGR-12	85	91	93.8	85	91.4
OGR-13	106	111	115.1	106	113.7
OGR-14	127	134	138.6	131	136.6
OGR-15	151	160	166.2	162	165.6
OGR-16	177	193	197.8	180	197.1
OGR-17	199	221	233.6	213	229.8
OGR-18	216	266	274	258	271.0
OGR-19	246	299	319	307	316.3

Fig. 5. Evolution of the best solution for OGR-13 in experiments with RK-EC alone and RK-EC with a heuristic. Results are averages of 30 runs

Fig. 6. Evolution of the best solution for OGR-18 in experiments with RK-EC alone and RK-EC with a heuristic. Results are averages of 30 runs

In graphs from figures 5 and 6 we show, respectively for OGR-13 and OGR-18, the evolution of the best solutions with both versions of the algorithms (RK-EC and RK-EC with heuristic). Results are averages of the 30 runs. These graphics confirm our analysis. Even though differences are always small, the solutions found by RK-EC with the heuristic are consistently better during the whole course of the simulation.

Although it is not possible to draw definite conclusions from these results, the small advantage provided by this simple heuristic suggest that maybe a more powerful method might further improve the search performance of the EC algorithm.

7 Conclusion

In this paper we presented a new EC algorithm used to search for good rulers for different OGR instances. Results achieved show that this evolutionary approach is effective since it was able to quickly discover good solutions. Best rulers found for all instances are clearly better than those ones achieved by the previous EC methods. Furthermore, we consider it as a realistic option to massive parallel approaches that need several months or years and a large computing power to discover high-quality Golomb rulers.

The proposed algorithm relies on RK to represent individuals from the population. One of the main advantages of this kind of representations is that they efficiently deal with permutations without requiring special precautions to build

initial populations or specific genetic operators that ensure that feasible individuals are generated. Another important feature of our representation is that it encodes redundant information (there are genes, i.e., segments, that are not used to build the ruler). The proposed interpretation algorithm is an extension of a previous one proposed by Rothlauf to deal with tree network representations.

We also analyzed the effect of a simple heuristic in the performance of the EC algorithm. Even though we proposed a straightforward heuristic we verified that, in most of the OGR instances, there was a small improvement in the results achieved (both in the best solutions found and in the averages of best solutions).

Results presented in this paper can be considered as preliminary. Our approach still evidences some scalability problems. Nevertheless, after confirming that an EC algorithm based on RK representation is a viable method to search for OGRs, we intend to extend our study in order to clarify some situations that were discussed in this paper. More precisely, our future research efforts will concentrate on two topics: (1) we will deepen our investigations related to representation of possible solutions and (2) we will try to develop new heuristics that are capable to provide a more efficient assistance to EC algorithms when searching for good solutions.

References

1. Golomb, S.: How to Number a Graph. In: Graph Theory and Computing. Academic Press (1972) 23–37
2. Bloom, G., Golomb, S.: Applications of numbered undirected graphs. In: Proceedings of the IEEE. Volume 65. (1977) 562–570
3. Dollas, A., Rankin, W., McCracken, D.: New algorithms for golomb ruler derivation and proof of the 19 mark ruler. IEEE Transactions on Information Theory **44** (1998) 379–382
4. Blum, E., Biraud, F., Ribes, J.: On optimal synthetic linear arrays with applications to radioastronomy. IEEE Transactions on Antennas and Propagation **AP-22** (1974) 108–109
5. Hayes, B.: Collective wisdom. American Scientist **86** (1998) 118–122
6. Klove, T.: Bounds and construction for difference triangle sets. IEEE Transactions on Information Theory **IT-35** (1989)
7. Bean, J.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing **6** (1994) 154–160
8. Soliday, S., Homaifar, A., G., L.: Genetic algorithm approach to the search for golomb rulers. In: Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95), Morgan Kaufmann (1995) 528–535
9. Dewdney, A.: Computer recreations. Scientific American (1985) 16–26
10. Gardner, M.: Mathematical games. Scientific American (1972) 198–112
11. Shearer, J.: Some new optimum golomb rulers. IEEE Transactions on Information Theory **IT-36** (1990) 183–184
12. Norman, B., Smith, A.: Random keys genetic algorithm with adaptive penalty function for optimization of constrained facility layout problems. In: Proceedings of the Fourth International Conference on Evolutionary Computation, IEEE (1997) 407–411

13. Rothlauf, F., Goldberg, D., Heinzl, A.D.: Network random keys - a tree representation scheme for genetic and evolutionary algorithms. Evolutionary Computation **10** (2002) 75–97
14. Magyar, G., Johnsson, G., Nevalainen, O.: An adaptive hybrid genetic algorithm fot the three-matching problem. IEEE Transactions on Evolutionary Computation **4** (2000) 135–146
15. Merz, P., Freisleben, B.: Genetic local search for the tsp: New results. In Back, T., Michalewicz, Z., Yao, X., eds.: Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE Press (1997) 159–164
16. Moscato, P.: Memetic Algorithms: a Short Introduction. In: New Ideas in Optimization. McGraw-Hill (1999) 221–234