Understanding the Role of Insertion and Correction in the Evolution of Golomb Rulers

Jorge Tavares, Francisco B. Pereira, Ernesto Costa
Evolutionary and Complex Systems Group
Centre for Informatics and Systems of the University of Coimbra
Polo II - Pinhal de Marrocos
3030 Coimbra, Portugal
{jast, xico, ernesto}@dei.uc.pt

Abstract— **An evolutionary algorithm designed to successfully search for Optimal Golomb Rulers is presented. The proposed approach uses a binary representation to codify the marks contained in a ruler. Standard genetic operators are used. During evaluation, insertion and correction procedures are applied in order to improve the algorithm performance. Experimental results show that this approach is effective and capable of identifying good solutions. Furthermore, a comprehensive study is performed to understand the role of insertion and correction. Results reveal that the first method is essential to the success of the search process, whereas the importance of the second one remains unclear.**

I. Introduction

Golomb Rulers are numerical sequences, or a class of undirected graphs, named after the relevant work of the mathematician Solomon Golomb [1], [2], that find application in a variety of engineering fields [3].

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 [4]. 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.

Fig. 1. A Perfect Golomb Ruler with 4 marks

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 [4].

As stated before, Golomb Rulers have application in a wide range of real world situations, mainly engineering. 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 [5], [6]. Other examples of areas where they are used include X-ray crystallography [2], or error detection and correction in the field of coding theory [4], [7].

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. Since this is an NP-hard problem [3], computation is prohibitively costly in terms of computer resources.

Evolutionary Computation (EC) approaches are a promising 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. Given its characteristics, evolutionary algorithms might be a competent choice to explore the search space and quickly identify good quality solutions. To the best of our knowledge, there are just three applications of EC to this problem [8], [9], [omitted reference]. The first two, when searching for solutions evolve the length of a fixed number of segments. This way, during search EC algorithms try to discover good rulers for a specific number of marks. In [omitted reference] we proposed a different evolutionary approach. Prior to the application of the algorithm, we specify the maximum ruler length and then the search procedure tries to determine how many marks can be placed in such a ruler as well as where each one of the marks should be located. The proposed approach relies on a binary representation and adopt standard genetic operators to carry out the exploration of the search space. Results presented show that this method is competitive. It was able to quickly discover good solutions and best rulers found are better than those discovered by previous EC approaches. In this paper we perform a detailed study on the role of the two procedures, insertion and correction, that are applied during the evaluation phase. With this analysis we aim to understand their influence the search process and under what circumstances they help to improve the algorithm performance.

The structure of the paper is the following: in section II we provide a formal definition of Golomb Rulers. Next, in section III, we present an overview of some of the main techniques used to generate and verify OGRs, including evolutionary approaches. Section IV comprises a description of our evolutionary model. Section V contains the presentation and analysis of the experimental results achieved. Finally, in section VI we draw some overall conclusions and suggest direction for future work.

II. Golomb Rulers: Representational Issues

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 a_n. 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 isn't any 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 (OGR-n, for short).

Finding OGRs is a complex combinatorial optimization problem. When designing an evolutionary algorithm to be used in this task, choosing an appropriate representation is one of the most important issues to consider. Following from the previous definitions, there are two options to contemplate:

- A candidate solution for a particular OGR-n instance is represented by a sequence of $n-1$ segment's lengths. According to this alternative, 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 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?
- 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 position where they should be placed.

In this paper we adopted the second approach. Details will be fully described in section IV.

III. 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. We present a brief description of some classical approaches, as well as the two evolutionary models developed for this problem.

A. Standard Approaches

One of the most important classes of algorithms proposed, aimed to construct OGRs from the scratch. Dewdney presented one of these methods in 1985 [10]. 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 [11] and Dollas et. al. [4] 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 [12] 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[1] and distributed.net[2]. 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 and 23 marks. The search for 24 and 25 marks started in July 2000.

Finally, Golomb Rulers are also used as case studies to analyze different Constraints Satisfaction models [13].

B. Evolutionary Approaches

The two previous attempts to apply evolutionary techniques to search for OGRs adopted representations based in permutations of segments.

In [8], Soliday et. al. proposed that an OGR-n candidate should be composed by a sequence 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

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

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

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 [9], another evolutionary approach based on the evolution of ruler segments is proposed. Contrary to the previous algorithm, a candidate solution for an OGR-n instance is composed by a permutation of λ distinct values, where λ is the maximum segment length.

Encoding of the permutation is done with Random Keys (RK) [14], [15], which ensures that the application of standard genetic operators (e.g., one point crossover) to chromosomes obtains feasible individuals. It is important to notice that the representation proposed in this approach is redundant, since a chromosome contains more segments than necessary to build the ruler. To obtain a possible solution (i.e., a ruler), a two-step process is performed: the RK encoding is translated into a permutation of integers and then an interpretation algorithm chooses the segments used in the ruler.

Evaluation of an individual follows two criteria: ruler length and legality of the solution (i.e., whether it contains repeated measurements). The effect of the addition of a simple heuristic to the interpretation process is also analyzed [9]. Results presented show a small improvement in the performance of the EC algorithm.

In table I we present the optimal length for some OGRs and compare them with the best results achieved by the described evolutionary approaches.

TABLE I
OGRS LENGTHS FOR INSTANCES BETWEEN 10 AND 16 MARKS AND BEST RESULTS ACHIEVED BY PREVIOUS EVOLUTIONARY APPROACHES

Instances	Optimal	Soliday EC	Random Keys	RK Heuristic
OGR-10	55	62	55	55
OGR-11	72	79	72	72
OGR-12	85	103	91	85
OGR-13	106	124	111	106
OGR-14	127	168	134	131
OGR-15	151	206	160	162
OGR-16	177	238	193	180

IV. Evolution with Marks

Following the discussion presented in section II, we propose a representation where candidate solutions encode information about the maximum number of marks that can be placed on a ruler.

A. Representation

We propose a straightforward chromosome representation that allow us to evolve both the number and position of marks for a given ruler with a predetermined maximum length L. 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 then it indicates that there is a mark in the corresponding position. This representation, where individuals represent sequences of marks in a ruler bounded by an upper length limit, enables us to search for rulers without a predetermined number of marks.

B. Genetic Operators

To generate descendants we selected variants of standard genetic operators usually applied to binary representations: two-point crossover, standard flip mutation and shift mutation. Shift mutation acts in the following way: it randomly selects a mark and shifts it left or right with equal probability. We selected this operator because it is somewhat sensitive to the structure of the solutions. Moving, deleting or inserting a mark can cause a great disruption in a ruler. Shift mutation is applied to all individuals. The number of shift mutations applied to each chromosome is given by a small constant S.

C. Evaluation

To evaluate an individual 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:

$$fitness(x) = \begin{cases} \text{- NR} & \text{, if } x \text{ is illegal} \\ \text{NM} \times \text{L + (L - M)} & \text{, if } x \text{ is legal} \end{cases} \quad (1)$$

Where *NR* is the number of repeated measurements, *NM* is the number of marks, L is the maximum length of the ruler and M is the length of the ruler encoded in the individual. 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 number of marks. Rulers with an equal number of marks are ranked according to their length.

A correction and insertion procedure is performed during the evaluation of an individual. The aim is twofold: 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, when checking the segments measured by the ruler, a duplicate measurement is found. A constant C gives the maximum number of rectifications allowed for a given solution. We use this upper bound so that the evaluation of an individual does not take too long. 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. The algorithm from figure 2 illustrates how this procedure works.

```
1. Check Ruler
2. While $count \leq C$ And $invalid$:
    2.1. Remove $mark$
    2.2. Check Ruler
    2.3. Increment $count$
3.If $valid$
    3.1. Add all possible marks
```

Fig. 2. Correction and Insertion phase

D. Previous Results

A preliminary set of results was presented in [omitted reference]. We used our evolutionary approach, which henceforth we will designate by Marks-EC, to seek for good rulers with lengths between 90 and 200 (given the known OGRs we looked for rulers between 12 and 16 marks). Marks-EC was able to find the OGR for all tested instances with the single exception for 15 marks. Nevertheless, for OGR-15 the solution discovered is just 2.65% higher than the optimal value, whilst the previous best value found by an evolutionary approach is 7.28% higher (155 versus 160). In table II we summarize the results achieved by our algorithm and compare them with previous evolutionary approaches.

TABLE II
BEST OGRS LENGTHS FOR INSTANCES BETWEEN 12 AND 16 MARKS ACHIEVED BY MARKS-EC IN COMPARISON WITH PREVIOUS APPROACHES AND OPTIMAL RESULTS

Instances	Optimal	Soliday EC	RK	RK Heuristic	Marks-EC
OGR-12	85	103	91	85	85
OGR-13	106	124	111	106	106
OGR-14	127	168	134	131	127
OGR-15	151	206	160	162	155
OGR-16	177	238	193	180	177

Instances with 15 and 16 marks revealed to be the most difficult. Golomb Rulers define search spaces whose topology is very hard to sample. The number of interactions that occur between the genes that compose a chromosome is one of the reasons that contribute to the difficulty of the search. There is a high epistasis associated with this problem, since changing the position of a mark affects all other marks. 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. In the next section we present an extended set of results that enable us to study the role played by the different components of the algorithm in the search process. Special attention is given to the insertion and correction procedures.

V. Experimental Results

To perform our analysis we focused our attention on a subset of the ruler lengths originally selected. More precisely, we used the evolutionary algorithm to seek for rulers with maximum lengths 100, 120, 170 and 200. Given the known OGRs these values enable Marks-EC to look for rulers with 12, 13, 15 and 16 marks respectively. The complete settings of the EC algorithm are the following: Number of evaluations: 15000000; Population size: 200; Tournament selection with tourney size: 5; Elitist strategy; Two-point crossover with rate 0.5; Mutation Operator: $\{$Flip, Shift$\}$; Number of shifts $S = \{0, 1, 2, 3\}$; Flip mutation rate: $\{0, 0.005, 0.01\}$; Maximum number of corrections, $C = \{0, 2, 4\}$; Insertion: $\{$Yes, No$\}$.

For every ruler length L we performed 30 runs with the same initial conditions and with different random seeds. All initial populations were randomly generated, where each gene will have a value of 0 with probability of 0.9 (to increase the probability of having some legal rulers in the initial population). 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. Each correction performed in an individual counts as one evaluation, since its fitness needs to be recalculated.

Tables IX to XIV present the overall results performed by us for our study. Column “L” refers to the maximum ruler length, “S” is the number of shifts, “n” is the number of marks of the best ruler found and “len” is the corresponding ruler length. Columns “avg” and “std” indicate the average and standard deviation of the best rulers found in each run. The evolutionary algorithm performed 30 runs for each configuration. In some cases the number of marks of the best ruler varies from run to run. When this occurred, the averages presented in the table reflect only the runs with the highest number of marks (column “runs”). For example, in table IX, with a mutation rate of 0.005, a maximum ruler length of 170, and with a shift size of 2, only for 17 runs were found rulers with 14 marks (the average and standard deviation take into account only those 17 runs). It wouldn’t be meaningful to compute all the runs. Since these tables contain a large amount of data, we will also present smaller tables with the most relevant outcomes. Tables with complete results are placed at the end of the paper.

A. The influence of Insertion

A brief perusal of the results 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 III we present a summary of the best rulers found without insertion. The algorithm was unable to find optimal rulers for any value of L; for larger values of L it couldn’t it couldn’t even obtain rulers with the desired number of marks. Comparing these results with those achieved with insertion, table IV, 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.

In terms of averages of the best solutions found for each of the 30 runs, the results are not very different. Table V and table VI present the averages with and without insertion respectively. Notice that table V reports some averages for

TABLE III

BEST RULERS FOUND WITHOUT INSERTION

L	C = 0 n	C = 0 len	C = 2 n	C = 2 len	C = 4 n	C = 4 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 IV

BEST RULERS FOUND WITH INSERTION

L	C = 0 n	C = 0 len	C = 2 n	C = 2 len	C = 4 n	C = 4 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

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 the 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. Furthermore, the experiments that were capable of attaining the maximum number of marks, the runs which actually achieve it were very few, being in most cases just a single one (as it can be seen from tables IX, XI and XIII).

TABLE V

SUMMARY OF AVERAGES WITHOUT INSERTION FOR THE BEST RULERS

L	C = 0 avg	C = 0 std	C = 2 avg	C = 2 std	C = 4 avg	C = 4 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

The previous reported effect, the number of times our approach could reach the maximum number of marks permitted by the maximum ruler length, L, is more perceptive in tables VII and VIII. 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 VIII, 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 VII).

TABLE VI

SUMMARY OF AVERAGES WITH INSERTION FOR THE BEST RULERS

L	C = 0 avg	C = 0 std	C = 2 avg	C = 2 std	C = 4 avg	C = 4 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

TABLE VII

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 VIII

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%

B. The role of Correction

We will concentrate now on the role of correction in the algorithm performance. Examining again tables VII and VIII, 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 EC 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.

From tables III and IV 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 IV, we verify that for settings of C and $L = \{100, 120\}$ the optimal rulers were discovered. For the remaining values of L, we see that the best result for $L = 170$ was obtained with $C = 4$, whilst 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 III 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 V and VI show us, independently of using insertion or not, for all levels of correction, the averages are all similar to each other. A closer inspection from table IX to table XIV confirms this analysis. These observations suggest that the value for C isn't relevant for the discovery of good solutions.

C. Shift and Flip Mutation

Before concluding our analysis, we report some observations regarding the operations of shift and standard flip mutation. While not a fundamental part of this study, it is possible to draw some conclusions about these genetic operators from the observation of tables X, XI, XIII and XIV. First of all, the utility of flip mutation does not seems to be important. In the presence of shift mutation, the lack of flip mutation does not bring significant changes in the results. On the other hand, there are some evidence that shift mutation is essential to the EC algorithm. All experiments made with $S = 0$ bear results a lot worse than those attained for the remaining values of S. The level of shift mutation, the S value, does not appear to have an effect on the EC algorithm performance. This indicates that the number of shift marks is irrelevant. In the absence of shift mutation there is some advantage in employing flip mutation. The attained results are slightly worse than the ones obtained with shift mutation. Nevertheless, the existence of flip mutation helps in the improvement of the algorithm performance.

VI. Conclusions and Future Work

An EC algorithm used to search for high quality Golomb Rulers is presented [Omitted Reference]. Previous achieved results show that this evolutionary approach is effective since it was able to quickly discover good solutions. The best rulers found are clearly better than those ones achieved by previous EC methods. We also attained known optimal values for most of the OGR tested instances.

The proposed approach uses a binary chromosome to represent potential Golomb Rulers. Each solution encodes both the number and the position of marks to be inserted in a blank ruler with a predetermined maximum length. One of the main advantages of this representation is that it does not require special precautions to build the initial population, nor specific genetic operators, to ensure that feasible individuals are generated. During the evaluation of the individuals, correction and insertion procedures are performed. The objective of these two procedures is to fix invalid rulers, and to see if it is possible to add marks to legal solutions, in order to improve the EC algorithm performance.

We performed a detailed study on the influence of insertion and correction. This analysis provides us some insights in under what circumstances these two procedures can help the algorithm to improve its performance. The role of insertion is fundamental in the discovery of good quality Golomb Rulers. Results show that when insertion is used, optimal or near optimal Golomb Rulers are found. This view is reinforced by the fact that without insertion, the algorithm has serious difficulties in producing good rulers. The influence of correction remains unclear. Apparently it produces no effects in the process of finding Golomb Rulers. It is also possible to conclude that shift mutation is essential to the algorithm, whereas the number of shifts is unimportant. Flip mutation seems to improve the performance of our approach when shift is not used. On the other hand, its role is not yet well established.

The results presented in this paper provide some insight in how the evolution of Golomb Rulers is affected by some aspects of our approach. Nevertheless, further studies are still required. First of all, additional scalability tests are required. It is necessary to establish the behavior of the EC algorithm for larger OGR instances (e.g., OGR-20 and above). Anyway, we consider this algorithm 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.

As a final remark, we were able to find a new lower bound for OGR-15 for an evolutionary approach: 153. Although not relevant to our study this is an interesting fact.

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

[4] 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

[5] 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

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

[7] Klove, T.: Bounds and construction for difference triangle sets. IEEE Transactions on Information Theory **IT-35** (1989)

[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] Pereira, F.B., Tavares, J., Costa, E.: Golomb rulers: The advantage of evolution. In: Proceedings of the 11th Portuguese Conference on Artificial Intelligence, Workshop on Artificial Life and Evolutionary Algorithms (ALEA), EPIA'03. (2003) 27–33

[10] Dewdney, A.: Computer recreations. Scientific American (1985) 16–26

[11] Gardner, M.: Mathematical games. Scientific American (1972) 198–112

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

[13] Smith, B.M., Stergiou, K., Walsh, T.: Using auxiliary variables and implied constraints to model non-binary problems. In: AAAI/IAAI. (2000) 182–187

[14] Bean, J.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing **6** (1994) 154–160

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

TABLE IX

RESULTS WITH C = 0 WITHOUT INSERTION

		Mutation 0.0					Mutation 0.005					Mutation 0.01				
L	S	n	len	avg	std	runs	n	len	avg	std	runs	n	len	avg	std	runs
100	0	10	77	91.91	6.47	11	11	82	92.20	4.91	30	11	82	89.50	4.27	30
	1	11	85	85.00	n/a	1	12	100	100.00	n/a	1	12	98	98.00	n/a	1
	2	11	87	65.42	n/a	1	12	99	99.50	0.71	2	11	83	88.03	2.51	30
	3	10	62	68.73	2.69	11	11	79	89.00	3.41	30	12	94	94.00	n/a	1
120	0	11	113	113.00	n/a	1	12	105	113.59	4.45	22	12	101	112.50	5.69	28
	1	11	91	99.38	6.63	13	12	101	109.97	3.75	30	12	99	112.79	5.16	29
	2	12	109	110.50	2.12	2	12	100	111.48	5.64	29	13	116	116.00	n/a	1
	3	11	89	93.00	2.71	4	12	105	115.12	3.88	25	12	103	114.00	5.33	23
170	0	12	153	161.50	5.92	6	14	157	164.83	4.63	12	14	154	164.00	4.19	18
	1	13	160	160.00	n/a	1	14	152	161.79	4.66	14	14	150	161.36	6.15	14
	2	12	111	121.00	5.31	12	14	153	164.59	4.35	17	14	151	164.14	6.49	7
	3	13	149	149.00	n/a	1	14	154	162.22	5.85	9	14	165	167.00	1.83	4
200	0	13	179	179.00	n/a	2	15	187	195.88	3.86	16	15	186	192.40	5.52	10
	1	13	157	165.90	5.67	10	15	174	189.00	8.90	6	15	199	199.00	n/a	1
	2	14	194	194.00	n/a	1	15	183	183.00	n/a	1	15	193	197.40	2.51	5
	3	13	135	153.20	13.07	5	15	192	195.33	3.06	3	15	177	186.67	10.60	3

TABLE X

RESULTS WITH C = 0 WITH INSERTION

		Mutation 0.0					Mutation 0.005					Mutation 0.01				
L	S	n	len	avg	std	runs	n	len	avg	std	runs	n	len	avg	std	runs
100	0	11	81	89.73	3.14	30	12	96	98.88	1.36	8	12	96	98.15	1.59	27
	1	12	85	89.90	2.63	30	12	85	90.77	1.22	30	12	85	89.87	2.54	30
	2	12	85	90.80	1.16	30	12	85	90.00	2.36	30	12	85	90.57	1.61	30
	3	12	85	90.17	2.12	30	12	85	90.37	1.92	30	12	85	90.60	1.99	30
120	0	12	106	111.97	2.97	30	12	102	106.77	2.97	30	13	113	117.21	9.58	19
	1	13	106	111.13	1.87	30	13	106	111.60	1.77	30	13	106	111.77	2.13	30
	2	13	109	111.33	1.27	30	13	109	111.70	1.21	30	13	109	112.27	1.39	30
	3	13	109	111.70	1.18	30	13	109	111.73	1.62	30	13	109	112.27	1.34	30
170	0	14	164	167.52	2.06	23	15	170	170.00	n/a	1	15	160	166.07	2.39	28
	1	15	155	163.27	2.92	30	15	160	164.87	1.81	30	15	161	165.03	1.79	30
	2	15	157	163.97	2.25	30	15	159	164.40	2.25	30	15	162	165.73	1.53	30
	3	15	159	163.97	2.06	30	15	162	164.67	1.63	30	15	160	165.07	2.24	30
200	0	15	186	194.46	4.10	13	15	170	184.97	5.17	30	16	190	196.86	3.12	22
	1	16	188	194.70	2.42	30	16	186	195.00	2.93	29	16	191	196.69	2.52	29
	2	16	189	194.80	2.47	30	16	191	195.80	2.43	30	16	188	196.44	2.85	27
	3	16	181	194.17	3.35	30	16	191	195.67	1.86	30	16	192	196.63	2.26	27

TABLE XI

RESULTS WITH C = 2 WITHOUT INSERTION

		Mutation 0.0					Mutation 0.005					Mutation 0.01				
L	S	n	len	avg	std	runs	n	len	avg	std	runs	n	len	avg	std	runs
100	0	10	81	90.38	5.64	13	11	82	92.50	4.65	28	12	99	99.00	n/a	1
	1	12	95	95.00	n/a	1	12	100	100.00	0.00	2	11	80	86.70	2.79	30
	2	10	65	70.67	2.63	18	11	81	88.73	2.91	30	12	99	99.00	n/a	1
	3	12	94	94.00	n/a	1	12	99	99.00	n/a	1	11	76	90.13	4.03	30
120	0	11	115	116.50	2.12	2	12	94	114.00	6.18	24	12	101	110.23	5.65	30
	1	12	102	112.61	5.34	28	12	102	110.83	4.31	30	12	99	114.15	5.25	27
	2	11	90	95.50	7.78	2	12	102	113.26	4.78	27	12	105	114.30	3.98	23
	3	12	104	115.00	4.97	19	12	101	112.33	6.29	21	12	109	114.72	3.29	18
170	0	12	144	156.25	11.70	4	14	159	164.56	3.71	9	14	152	161.96	5.15	23
	1	14	159	165.20	4.16	10	14	151	162.27	7.06	15	14	149	160.67	6.39	15
	2	12	121	124.29	2.93	7	14	156	165.00	5.14	6	14	161	165.38	3.46	8
	3	14	152	162.00	5.87	5	14	167	167.00	n/a	1	14	166	168.00	2.83	2
200	0	13	177	180.00	2.65	3	15	184	193.13	5.49	8	15	186	181.63	5.60	14
	1	15	190	194.33	5.13	3	15	194	196.33	1.37	6	15	188	193.57	4.16	7
	2	13	145	158.33	9.24	6	15	179	190.67	8.04	6	15	192	196.50	3.42	4
	3	15	195	195.00	n/a	1	15	200	200.00	n/a	1	14	165	183.42	10.48	24

TABLE XII

RESULTS WITH C = 2 WITH INSERTION

		Mutation 0.0					Mutation 0.005					Mutation 0.01				
L	S	n	len	avg	std	runs	n	len	avg	std	runs	n	len	avg	std	runs
100	0	12	97	98.33	1.15	3	12	96	98.57	1.60	14	12	92	96.61	2.10	28
	1	12	85	90.97	1.38	30	12	85	90.43	1.91	30	12	85	90.80	1.27	30
	2	12	85	90.43	1.55	30	12	85	90.83	1.26	30	12	85	90.17	2.44	30
	3	12	85	90.50	1.96	30	12	85	90.43	2.28	30	12	85	90.53	2.01	30
120	0	13	119	119.00	n/a	1	13	115	117.50	2.08	4	13	114	117.67	1.69	24
	1	13	109	112.03	1.50	30	13	109	111.30	1.29	30	13	106	111.87	1.96	30
	2	13	109	111.53	1.22	30	13	106	111.37	1.94	30	13	106	112.13	2.32	30
	3	13	109	112.63	1.45	30	13	106	111.47	1.78	30	13	110	112.80	1.42	30
170	0	14	157	165.88	3.93	25	15	165	167.00	2.83	2	15	161	166.86	2.37	29
	1	15	162	166.23	2.05	30	15	161	165.03	2.06	30	15	162	166.53	1.96	30
	2	15	160	164.20	1.67	30	15	162	165.43	1.52	30	15	162	166.13	1.72	30
	3	15	159	165.70	2.38	30	15	159	164.67	2.66	30	15	159	165.87	2.45	30
200	0	16	200	200.00	n/a	1	16	196	197.00	1.41	2	16	189	196.62	3.01	21
	1	16	193	197.07	1.88	27	16	190	196.33	2.75	30	16	194	197.04	2.03	26
	2	16	188	195.75	2.82	28	16	192	196.48	2.31	29	16	194	197.42	1.77	26
	3	16	190	197.30	2.20	27	16	189	197.00	2.69	28	16	186	196.44	2.97	27

TABLE XIII

RESULTS WITH C = 4 WITHOUT INSERTION

		Mutation 0.0					Mutation 0.005					Mutation 0.01				
L	S	n	len	avg	std	runs	n	len	avg	std	runs	n	len	avg	std	runs
100	0	10	78	91.46	5.65	13	11	78	91.07	5.33	29	11	82	90.87	4.68	30
	1	10	71	76.68	4.11	19	11	78	87.03	4.16	30	11	82	88.23	3.26	30
	2	11	96	65.05	n/a	1	11	84	92.27	2.64	30	12	100	100.00	n/a	1
	3	10	68	70.70	1.89	10	11	86	92.89	3.39	28	11	82	92.37	4.52	30
120	0	11	108	115.00	4.12	5	12	105	113.91	3.66	22	12	100	111.40	5.05	30
	1	11	88	100.00	6.63	10	12	99	110.52	5.87	25	12	97	113.38	5.73	24
	2	11	89	94.40	4.45	5	12	108	114.45	3.33	20	12	99	111.24	6.44	21
	3	11	87	93.33	7.09	3	12	101	111.57	5.98	21	12	102	113.13	5.38	16
170	0	12	151	159.20	7.12	5	14	157	164.00	4.22	10	14	151	164.16	4.52	25
	1	13	142	146.50	6.36	2	14	155	163.73	4.43	11	14	158	164.80	5.45	5
	2	13	128	128.00	n/a	1	14	163	165.00	1.58	5	14	160	164.00	3.56	4
	3	13	154	154.00	n/a	1	14	165	165.00	n/a	1	14	154	159.75	6.45	4
200	0	13	185	190.00	7.07	2	15	188	193.50	4.23	6	15	185	191.60	4.93	10
	1	13	158	174.63	8.26	8	15	181	192.00	7.39	4	15	191	194.00	4.24	2
	2	14	185	185.00	n/a	1	15	193	197.25	3.10	4	15	194	194.00	n/a	1
	3	14	183	183.00	n/a	1	14	161	185.38	11.66	24	14	162	185.07	9.38	15

TABLE XIV

RESULTS WITH C = 4 WITH INSERTION

		Mutation 0.0					Mutation 0.005					Mutation 0.01				
L	S	n	len	avg	std	runs	n	len	avg	std	runs	n	len	avg	std	runs
100	0	12	96	98.00	2.83	2	12	91	97.74	7.25	27	12	90	95.87	2.03	30
	1	12	85	91.47	1.59	30	12	85	90.30	1.86	30	12	85	90.07	2.36	30
	2	12	90	90.87	0.68	30	12	85	89.93	2.32	30	12	85	90.13	2.16	30
	3	12	85	90.20	1.86	30	12	85	90.97	1.35	30	12	85	90.57	1.98	30
120	0	13	118	118.00	n/a	1	13	116	119.00	1.11	14	13	110	116.93	2.63	27
	1	13	109	112.57	1.92	30	13	106	111.20	1.63	30	13	109	112.20	1.40	30
	2	13	109	111.03	1.33	30	13	109	111.40	1.30	30	13	109	112.60	1.48	30
	3	13	109	111.40	1.38	30	13	109	112.57	1.38	30	13	109	112.13	1.33	30
170	0	14	159	163.89	2.63	28	15	170	170.00	n/a	1	15	163	167.48	1.76	25
	1	15	155	163.96	3.24	28	15	158	164.53	2.69	30	15	153	165.27	3.15	30
	2	15	157	163.77	2.47	30	15	153	165.60	3.47	30	15	160	165.93	1.87	30
	3	15	155	164.70	2.93	30	15	159	165.47	2.18	30	15	155	166.03	2.59	30
200	0	15	187	196.21	3.58	24	15	173	178.37	2.86	30	16	193	197.83	1.90	23
	1	16	190	195.64	2.68	22	16	183	195.77	3.30	30	16	186	196.50	3.66	26
	2	16	190	195.97	2.54	29	16	191	197.14	2.27	28	16	194	197.28	2.15	25
	3	16	189	196.00	3.20	29	16	188	196.64	2.42	28	16	189	196.92	2.74	25