Evolving Segments Length in Golomb Rulers

Jorge Tavares[1], Tiago Leitão[1], Francisco B. Pereira[1,2], Ernesto Costa[1]
[1]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
E-mail: {jast, xico, ernesto}@dei.uc.pt, tleitao@student.dei.uc.pt

Abstract

An evolutionary algorithm based on Random Keys to represent Golomb Rulers segments, has been found to be a reliable option for finding Optimal Golomb Rulers in a short amount of time, when comparing with standard methods. This paper presents a modified version of this evolutionary algorithm where the maximum segment length for a Golomb Ruler is also part of the evolutionary process. Attained experimental results shows us that this alteration does not seems to provide significant benefits to the static version of the algorithm.

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], 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. Figure 1 is an example of a Golomb Ruler.

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. There may exist multiple different OGRs for a specific number of marks. 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 [2].

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. There have been some applications of EC to this problem [3], [4], [5] and [6]. These approaches, 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 [7] a different evolutionary approach is proposed. Prior to the application of the algorithm, a maximum ruler length is specified 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. In this paper we continue the study presented in [5], by analyzing the influence of evolving the maximum length of a segment.

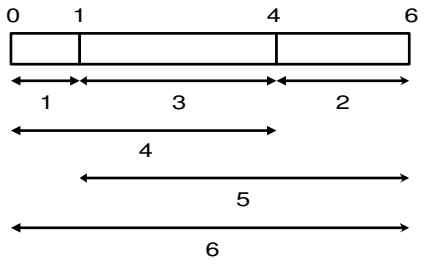

Fig. 1. A Golomb Ruler with 4 marks

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 an. In such a case an 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 [3]. 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?

3 An Evolutionary Approach with Random Keys

The representation chosen for individuals plays a crucial role on the performance of EC algorithms. In [5], an evolutionary approach based on the evolution of ruler segments is proposed. A candidate solution for an OGR-n instance is composed by a permutation of λ distinct values, where λ is the maximum segment length.

Even when 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.

In our approach, the chromosome is composed by a permutation of λ distinct values. Encoding of the permutation is done with random keys (RK). RKs were introduced by Bean [8] in 1994 and obtained good results in situations where the relative order of the tasks is important [8], [9]. 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 RKs. For a detailed description, consult [8] or [10]. RKs 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 [10], Rothlauf et. al. proposed NetKeys, an extension of RKs 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 RKs. 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

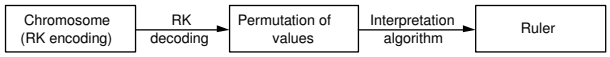

Fig. 2. Decoding and interpretation of the information contained in a chromosome

relative position. Figure 2 illustrates how decoding and interpretation (the two stages required to assign fitness to an individual) are related.

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. 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.

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 [5]. Results presented show a small improvement in the performance of the EC algorithm.

In this paper, the value for λ is also evolved. This is accomplished in a simple manner. The value for λ is given by the chromosome length, this means that by having chromosomes with variable length each individual will have a different λ value. The only modification that is necessary to the previous approach, with fixed length, is made on the crossover operator, that for each individual, a different cut point is randomly selected. This will ensure the swap of genetic material of different lengths, thus providing individuals with variable length.

4 Experimental Results

To evaluate our approach 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 17 marks. The settings of the EC algorithm are the following: Number of generations: 5000; Population size: 100; Tournament selection with tourney size: 5; Elitist strategy; One point crossover with rate: 0.75; Since we are mainly interest in comparing both random keys approaches, the number of total evalutions is smaller than the usual number needed for attaining the best results as presented in [5].

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:

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

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 0.05.

Table 1. Best rulers found for 30 runs without the use of a simple heuristic and comparing with optimal results.

Instances	Optimal	RK Fix λ	RK Evolve λ
OGR-10	55	55	55
OGR-11	72	72	72
OGR-12	85	91	91
OGR-13	106	111	114
OGR-14	127	131	131
OGR-15	151	167	167
OGR-16	177	200	202
OGR-17	199	236	230

The examination of table 1 shows that both RK approaches have found good quality solutions when comparing to the known optimal results. As expected, for a smaller number of marks, both RK approaches were able to find optimal solutions, while for the larger instances, the rulers found are of poorer quality. Nevertheless, it is interesting to observe that are not great differences between both RK approaches. As a matter of fact, the best solutions found are always the same with exception for the instances with 13, 16 and 17 marks. For OGR-12 and OGR-15, the static version of the algorithm provide better solutions, while for OGR-17 evolving λ has attained a better solution.

The observation of the best rulers found is not a sufficient indication if the introduction of the evalution of λ is a worthwhile addition to the algorithm. By looking

Table 2. Best rulers found and averages of the best rulers for 30 runs, for all the tested approaches without the use of a simple heuristic.

Instances	RK Fix λ		RK Evolve λ		t-test
	best	avg	best	avg	$p \leq 0.05$
OGR-10	55	60.1	55	**57.8**	**0.000159**
OGR-11	72	76.0	72	**74.9**	**0.031262**
OGR-12	91	95.6	91	**94.4**	**0.008301**
OGR-13	**111**	**117.2**	114	118.5	**0.017769**
OGR-14	131	**143.6**	131	144.4	0.672176
OGR-15	167	**172.8**	167	174.6	0.054341
OGR-16	**200**	**207.8**	202	210.5	**0.008406**
OGR-17	236	**245.5**	**230**	246.5	0.157805

Table 3. Best rulers found and averages of the best rulers for 30 runs, for all the tested approaches with the use of a simple heuristic.

Instances	RK Fix λ		RK Evolve λ		t-test
	best	avg	best	avg	$p \leq 0.05$
OGR-10	55	**58.0**	55	58.1	0.953500
OGR-11	**72**	**75.0**	74	75.5	0.130064
OGR-12	**92**	**94.5**	95	95.3	**0.001153**
OGR-13	113	**116.4**	**112**	118.0	**0.014721**
OGR-14	**137**	**142.1**	137	144.2	**0.001120**
OGR-15	**167**	**171.5**	169	173.3	**0.045272**
OGR-16	**197**	**205.0**	198	206.8	0.094827
OGR-17	**228**	**240.6**	234	243.5	**0.021833**

at table 2, we can also compare the averages of the best solutions found for the 30 runs (column avg). Results in bold indicates better results as a pattern is formed: for smaller instances, the evolution of λ consistently finds higher quality rulers while for a larger number of marks, not evolving λ attains overall better averages. Making a statistical analysis of the results, we can find significant differences for most of the OGR instances (column t-test, significant results in bold).

A final experiment was performed regarding this issue. Table 3 presents the attained results with both RK approaches with the usage of a simple heuristic (as in [5]). The use of an heuristic favouring small segments introduced another difficult factor for the evolution of the maximum segment length. In terms of averages, the static version is always better and for best rulers found (with the single exception of OGR-13).

5 Conclusions

In this paper we presented some experiments regarding the evolution of the maximum segment length of a Golomb Ruler. This effect was attained by introducing variable length cromosomes. Results presented in this paper suggest that there aren't significant gains in evolving the maximum segment length. Even so, it seems that there might be some advantages for lower instances.

References

[1] Golomb, S.: How to Number a Graph. In: Graph Theory and Computing. Academic Press (1972) 23–37

[2] 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

[3] 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

[4] Feeney, B.: Determining optimum and near-optimum golomb rulers using genetic algorithms. Master's thesis, University College Cork (2003)

[5] 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

[6] Cotta, C., Fernndez, A.J.: A hybrid grasp - evolutionary algorithm approach to golomb ruler search. In: Proceedings of Parallel Problem Solving from Nature (PPSNVIII). (2004)

[7] Tavares, J., Pereira, F.B., Costa, E.: Understanding the role of insertion and correction in the evolution of golomb rulers. In: Proceedings of the 2004 Congress on EC (CEC04). (2004) 69–76

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

[9] 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

[10] 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