A Hybrid GRASP – Evolutionary Algorithm Approach to Golomb Ruler Search

Carlos Cotta and Antonio J. Fernández

Dept. Lenguajes y Ciencias de la Computación, ETSI Informática,
University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain
{ccottap,afdez}@lcc.uma.es

Abstract. We consider the problem of finding small Golomb rulers, a hard combinatorial optimization task. This problem is here tackled by means of a hybrid evolutionary algorithm (EA). This EA incorporates ideas from greedy randomized adaptive search procedures (GRASP) in order to perform the genotype-to-phenotype mapping. As it will be shown, this hybrid approach can provide high quality results, better than those of reactive GRASP and other EAs.

1 Introduction

The concept of Golomb rulers was first introduced by W.C. Babcock in 1953 [1], and further described by S.W. Golomb [2]. Golomb Rulers are a class of undirected graphs that, unlike usual rulers, measure more discrete lengths than the number of marks they carry. The particularity of Golomb Rulers is that on any given ruler, all differences between pairs of marks are unique. This feature makes Golomb Rulers really interesting in many practical applications (e.g., [3, 4] explain the connection of Golomb Rulers to real world problems in diverse fields such as radio communications, X-ray crystallography, coding theory, radio astronomy, and pulse phase modulation communications).

Traditionally, researchers are usually interested in discovering rulers with minimum length and Golomb rulers are not an exception. An *Optimal Golomb Ruler* (OGR) is defined as the shortest Golomb ruler for a number of marks. There may be multiple OGRs for a specific number of marks.

The search for OGRs is an extremely difficult task as it is a combinatorial problem whose bounds grow geometrically with respect to the solution size [5]. This has been a major limitation as each new ruler to be discovered is by necessity larger than its predecessor. Fortunately, the search space is bounded and, therefore, solvable [6]. To date, the highest Golomb ruler whose shortest length is known is the ruler with 23 marks [7, 8]. Best solutions for OGRs with a number of marks between 20 and 23 were obtained by massive parallelism projects, and it took several months to find optimum for each of those instances [9, 4]. In July 2000, distributed.net started the web search for the 24 and 25 OGR's, although no news exists about the success in finding these optimums.

To our knowledge, there have been three attempts to apply evolutionary algorithms (EAs) to the search for OGRs (see section 2.2). In this paper, we present a hybrid evolutionary approach designed to speed-up the process of searching good solutions. This approach is based on using ideas taken from the GRASP metaheuristic [10]. To be precise, we propose the use of a GRASP-like mechanism to perform the genotype-to-phenotype mapping.

2 Background

The OGR problem can be classified as a fixed-size subset selection problem, such as e.g., the p-median problem [11]. It exhibits some very distinctive features though. A brief overview of the problem, and how it has been tackled in the literature is provided below.

2.1 Golomb Rulers

A *n-mark Golomb ruler* is an ordered set of n distinct non-negative integers, called *marks*, $a_1 < ... < a_n$, such that all the differences $a_i - a_j$ $(i > j)$ are distinct. Clearly we may assume $a_1 = 0$. By convention, a_n is the *length of the Golomb ruler*. A Golomb ruler with n marks is an optimal Golomb ruler if, and only if,

– there exists no other n-mark Golomb rulers having smaller length, and
– the ruler is canonically "smaller" with respect to the the equivalent rulers. This means that the first differing entry is less than the corresponding entry in the other ruler.

Fig. 1 shows an OGR with 4-marks. Observe that all distances between any two marks are different.

Typically, Golomb Rulers are represented by the values of the marks on the ruler, i.e., in a n-mark Golomb ruler, $a_i = x$ $(1 \leqslant i \leqslant n)$ means that x is the mark value in position i. The sequence $(0, 1, 4, 6)$ would then represent the ruler in Fig. 1. However, this representation turns out to be inappropriate for EAs (for example, it is problematic with respect to developing good crossover operators [12]). An alternative representation consists of representing the Golomb ruler via the lengths of its segments, where the length of a segment of a ruler is defined as the distance between two consecutive marks. Therefore, a Golomb Ruler can

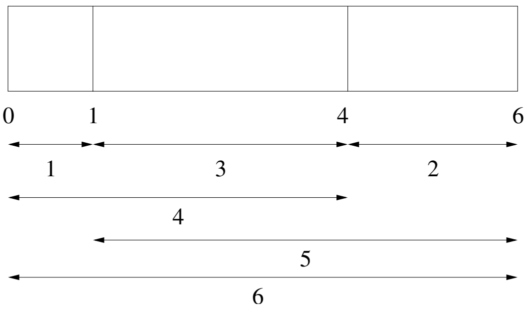

Fig. 1. A Golomb Ruler with 4 marks.

be represented with $n-1$ marks specifying the lengths of the $n-1$ segments that compose it. In the previous example, the sequence $(1,3,2)$ would encode the ruler depicted in Fig. 1.

2.2 Evolutionary Approaches to the OGR

Golomb rulers have been solved by using very different techniques. We will restrict here just to the evolutionary approaches considered so far. The interested reader on other techniques is referred to [7, 8].

To our knowledge, there have been three attempts to apply Evolutionary Computation (EC) techniques to the search for OGRs. In 1995, Soliday, Homaifar and Lebby [12] used a genetic algorithm on different instances of the Golomb Ruler problem. In their representation, each chromosome is composed by a permutation of $n-1$ integers that represents the sequence of the $n-1$ lengths of its segments. In order to assure the uniqueness of each segment length in all the individuals of the population, certain precautions were taken: initially, an array with numbers 1 to m (i.e. the maximum segment length) was loaded; then a repetitive sequence of random swaps between two positions i,j ($i \neq j$ and $i,j > 1$) in the array was executed; finally the first $n-1$ are selected and position 1 (initially containing the value 1) was randomly swapped with some of those selected positions. Soliday *et al.* also prevented mirror image representations by aligning the ruler. Two evaluation criteria were followed: the overall length of the ruler, and the number of repeated measurements. The mutation operator consisted of two types: a permutation in the segment order, or a change in the segment lengths. As with the population generation, special precautions were taken to assure that segment of length 1 is retained in the ruler as it was proved that all good Golomb rulers should have a segment of length one [2]. A special crossover operator was designed to guarantee that descendants are valid permutations and that length 1 is also retained.

Later, Feeney studied the effect of hybridizing genetic algorithms with local improvement techniques to solve Golomb rulers [3]. Namely, Feeney examined three methods of searching for Golomb Rulers, using genetic algorithms on its own, with local search and Baldwinian learning, and with local search and Lamarckian learning (see e.g. [13–15] for more information on Lamarckian and Baldwinian learning). It is known that, combined with EAs, local search techniques often reduces drastically the number of generations to find a near-optimum solution (see e.g., [16]). However, this can be also a weakness since it can result in premature convergence to suboptimal solutions in certain problems. Moreover, distinct learning techniques may behave differently in solving the same problem; therefore, these techniques tend to depend greatly on the problem itself. The representation used consisted of an array of integers corresponding to the marks of the ruler. The crossover operator was similar to that used in Soliday *et al.*'s approach although a sort procedure was added at the end. Mutation process was applied on each segment of the ruler with a mutation probability, and consisted basically in the addition to the segment mark selected for mutation of a random amount in the range $[-x,x]$ where x is the maximum difference between any pair of marks in any ruler of the initial population.

Table 1. Results obtained by Soliday *et al.*'s and Feeney's approaches. The columns denoted by Len, Pob, Gen and Err, respectively indicate the length of the best solution calculated, the initial population size, the number of generations required to produce the reported length, and the relative error with respect to the shortest known length (shown in column opt).

		Soliday *et al.*				Feeney			
Instances	opt.	Len	Pob	Gen	Err	Len	Pob	Gen	Err
OGR-5	11	11	512	10	0%	11	100	6	0%
OGR-6	17	17	512	22	0%	17	100	6	0%
OGR-7	25	25	512	22	0%	25	100	23	0%
OGR-8	34	35	1024	84	2.94%	34	100	18	0%
OGR-9	44	44	1024	433	0%	47	100	166	6.81%
OGR-10	55	62	1024	458	12.73%	62	100	229	12.72%
OGR-11	72	79	1024	152	9.72%	80	100	166	11.11%
OGR-12	85	103	1024	59	21.18%	101	100	138	18.82%
OGR-13	106	124	1024	664	16.98%	122	100	128	15.09%
OGR-14	127	168	2048	1506	32.28%	146	100	135	14.96%
OGR-15	151	206	2048	858	36.42%	281	100	79	18.86%
OGR-16	177	238	2048	708	36.46%	213	100	48	20.33%

Table 1 shows the results produced by Soliday *et al.*'s and Feeney's approaches. The best known OGR for every instance size have been taken from [7, 8]. With respect to the different approaches studied by Feeney, we have selected the best result obtained in [3], that corresponds to the execution of a genetic algorithm (with high mutation and no crossover) without local search. The maximum number of generations allowed was 250; the program was executed three times and the most typical result[1] found is shown. These results indicate that Soliday *et al.*'s approach is not very good compared to further EA approaches to solve the Golomb problem. Observe that, for rulers with 10 to 16 marks, the relative error is far from the known OGRs as it is between 9.72% and 36.46%. On its turn, Feeney's approach behaves better and maintains, for the same rulers, this error between 6.81% and 20.33%, that means a significant improvement.

The hybrid approach of Feeney was found to be successful in generating near optimum rulers and in generating optimal Golomb rulers of a short length. Observe also that the error remains in the short range [14.96%,20.33%] for rulers with marks from 12 to 16. This clearly indicates a stabilization of the error. However, as it is reported in [3], performance was really poor, as it was expected, mainly due to the local search.

Recently, Pereira *et al.* [17] have presented a new EA approach to find OGRs. This new approach uses the concept of random keys [18] to codify the information contained in each chromosome. As in the Soliday *et al.*'s approach, any candidate solution for a n-mark OGR is represented with the length of each of its $n-1$ segments. In fact, a chromosome is composed by a permutation

[1] The most typical result is the ruler whose length is closest to the average of the three, unless two had the same length in which case it was their length.

of λ distinct values (where λ is the maximum segment length), and encoding of the permutation is done with random keys (RK). The basic idea consists of generating n random numbers (i.e., the keys) sampled from the interval $[0, 1]$ and ordered by its position in the sequence $1, \ldots, n$; then all positions of the keys are sorted, in decreasing order, according to the value that they contain. This guarantees to obtain always a permutation of n unique numbers between 1 and n. The codification uses a more advanced principle based on the concept of NetKeys (an extension of RK to problems dealing with tree network design) although the basic idea is that described above. Two evaluation criteria, similar to those described in [12], were followed: ruler length and whether the solution contains repeated measurements. They also presented an alternative algorithm that adds a heuristic, favoring the insertion of small segments, to the RK approach already proposed. We will return later to this approach since it has been included in our experimental comparison.

3 A GRASP-Based Hybrid Approach

As anticipated in Sect. 1, the EA approach proposed is based on incorporating ideas from GRASP. It is thus necessary discussing firstly the deployment of this latter metaheuristic on the OGR problem.

3.1 Basic GRASP for the OGR Problem

Greedy randomized adaptive search procedures can be viewed as repetitive sampling techniques [19]. Each iteration of the algorithm produces a tentative solution for the problem at hand by means of a greedy randomized construction algorithm. This latter algorithm assumes that solutions are represented by a list of attributes, and builds incrementally solutions by specifying values for these attributes. More precisely, the values for each attribute are ranked according to some local quality measure, and selected using a quality-biased mechanism. The pseudocode of this process would be as follows:

```
1. $sol \leftarrow \emptyset$
2. **while** $\neg$completed($sol$) **do**
   (a) $RCL \leftarrow$ build a ranked candidate list.
   (b) $s \leftarrow$ select attribute value from $RCL$.
   (c) $sol \leftarrow sol \cup \{s\}$
3. **return** $sol$
```

One of the key steps in this pseudocode is the selection of an attribute from RCL. This can be typically done by using a qualitative criterion (i.e., a candidate is selected among the best k elements in RCL, where k is a parameter), or a quantitative criterion (i.e., a candidate is selected among the elements whose quality is between q_1 and $q_1 + \alpha \cdot (q_{|RCL|} - q_1)$, where q_i is the quality of the i-th element of RCL and α is a parameter). Notice that having $k = 1$ or $\alpha = 0$ would thus result in a plain greedy heuristic. GRASP is based on iterating this construction procedure (possibly applying some local improvement technique to the so-obtained solutions), keeping the best solution generated along the run.

In the OGR problem, the attributes of solutions are obviously the position of the marks. The construction procedure would then iteratively place each of the $n-1$ marks (the first mark is assumed to be $a_1 = 0$). The $(i+1)$-th mark can be obtained as $a_{i+1} = a_i + l_i$, where $l_i \geqslant 1$ is the i-th segment length. We thus have a potential list of candidates based on tentative values for $l_i \in \{1, 2, \cdots\}$. Actually, this potential list of candidates can be as long as desired, although a bound $\lambda \in O(n)$ is typically chosen. Of course, many candidates from this potential list are infeasible since they would lead to repeated distances between marks. A potential value l_i would then be feasible if, and only if, for all j, k, r such that $1 \leqslant j \leqslant i$ and $1 \leqslant k < r \leqslant i$, it holds that $(a_i + l_i - a_j) \neq (a_r - a_k)$. After filtering infeasible segment lengths (only values $l_i \leqslant \max_{1 \leqslant k < r \leqslant i}(a_r - a_k)$ have to be checked) we come up with the elements of the actual RCL. A quality measure is now required. In this problem, the natural measure is the length of the ruler. Since at each step the value of a_i is known, it turns out that the ranked list consists of the sequence of increasing feasible values for l_i. A qualitative selection criterion can then be defined by picking a random candidate among the smallest k feasible values for l_i.

3.2 Reactive GRASP vs. Hybrid EA

One of the potential problems of the basic GRASP procedure described before relies on the selection of the parameter for selecting an attribute value from the *RCL*. As shown in [20], using a single fixed value for this parameter may hinder finding high-quality solutions. Several options are possible to solve this problem. In particular, a learning-based strategy termed *reactive* GRASP has been proposed [21]. In this case, the value of the parameter is chosen in each iteration from a set of discrete values $\Pi = \{\pi_1, \cdots, \pi_m\}$. The selection of the precise value of the parameter at each iteration can be done on the basis of the goodness γ_i of the best solution ever generated by using each parameter π_i. Any of the selection mechanisms typically used in EAs can be utilized for this purpose. For example, in [21], a roulette-wheel procedure is proposed. Since we are dealing here with a minimization problem, such a proportional approach would not be possible unless goodness values were appropriately transformed. We have opted for a simpler approach: using a non-proportionate approach. To be precise, we have considered binary tournament for selecting parameter values.

Using a reactive approach allows the algorithm focusing on the more appropriate subset of parameter values. However, it can still face difficulties if optimal (or near-optimal) solutions comprise an attribute value whose rank in the *RCL* is high: a low value of the selection parameter would preclude picking this attribute value; a high enough value of the selection parameter could select it, but many other low-quality attributes as well. A finer-grain mechanism would be required here, an approach that allowed for using different values of the parameter not just in each application of the construction phase, but in each internal step of the construction algorithm. This is where EAs come into play.

EAs can be used to evolve the sequence of $n-1$ selection parameters used within an application of the construction algorithm. In principle, this implies that each individual would be a sequence $\langle r_1, \cdots, r_{n-1} \rangle$, where r_i would be the parameter used in the i-th iteration of the construction algorithm. Two practical consideration must be taken into account though. The first one refers to the genotype-to-phenotype mapping: by making randomized choices of attribute values this mapping would be stochastic. Since this would result in an increased level of complexity of the algorithm, a deterministic choice is made. To be precise, the value r_i indicates that the r_i-th best attribute value should be selected in the i-th step. The second consideration refers to the last step of the construction algorithm. In this last step it does not make sense to pick any other attribute value than the smallest one. For this reason, $r_{n-1} = 1$, and individuals need only contain the sequence $\langle r_1, \cdots, r_{n-2} \rangle$. Notice that this representation of solutions is orthogonal [22], i.e., any sequence represents a feasible solution, and hence, standard operators for crossover and mutation can be used to manipulate them.

4 Experimental Results

The experiments have been performed using four different algorithms: plain GRASP, reactive GRASP, a permutational EA following [17], and the hybrid EA-GRASP approach described in previous section. The plain GRASP algorithm used a qualitative selection mechanism using $k = n$ as parameter. In the case of reactive GRASP, five different equally-spaced values between 2 and n were considered. As to the EAs, an elitist generational model ($popsize = 100$, $p_X = .9$, $p_M = 1/n$) with binary tournament selection has been utilized. For the hybrid EA, each gene can take values $r_i \in \{1, \cdots, n\}$, uniform crossover is used, and mutation is done by randomly increasing or decreasing a gene by 1. The permutational EA uses the interpretation mechanism described in [17]. Random keys are here directly substituted by permutations, being PMX used for crossover, and the swap operator for mutation. In all cases, the algorithms have been run 30 times for 10^6 evaluations. No fine tuning of these parameters has been attempted.

The results of plain GRASP and reactive GRASP are shown in Table 2. As it can be seen, reactive GRASP quickly outperforms plain GRASP as the instance size increases. Notice also that the results of this reactive GRASP are better than those of the basic EA approaches reported in Sect. 2.2.

Focusing on the EAs, the results are shown in Table 3. Notice firstly that the permutational EA provides comparable results to those of reactive GRASP. The hybrid EA provides roughly the same performance that the permutational EA for small instance sizes, but becomes clearly superior when the number of marks increases. A non-parametrical statistical test – Wilcoxon ranksum – indicates that the differences are significant for 10, 11, 14, 15, and 16 marks.

5 Conclusions

We have presented a hybrid EA that incorporates a GRASP-like procedure for decoding the chromosome into a feasible solutions. The advantages of this approach are twofold: first of all, problem-knowledge is exploited by means of the

Table 2. Results (averaged for 30 runs) of plain GRASP and reactive GRASP.

	Plain GRASP			
	ruler length			evaluations
instance	best	median	mean $\pm\sigma$	mean $\pm\sigma$
OGR-5	11	11	11.0 ± 0.0	44.0 ± 45.0
OGR-6	17	17	17.0 ± 0.0	128.7 ± 106.9
OGR-7	25	25	25.0 ± 0.0	1634.0 ± 1243.2
OGR-8	34	34	34.0 ± 0.0	202710.6 ± 230112.6
OGR-9	44	44	44.8 ± 0.9	377499.8 ± 204172.4
OGR-10	60	62	61.7 ± 0.8	546206.5 ± 226282.5
OGR-11	78	81	80.8 ± 1.6	443928.7 ± 204992.1
OGR-12	103	105	104.8 ± 1.3	478968.6 ± 304395.1
OGR-13	127	129	129.9 ± 2.2	641836.0 ± 180381.4
OGR-14	148	163	161.9 ± 5.1	645870.3 ± 226777.6
OGR-15	191	200	199.1 ± 4.3	380212.7 ± 313521.6
OGR-16	226	242	242.1 ± 6.1	463089.9 ± 237976.0
	Reactive GRASP			
	ruler length			evaluations
instance	best	median	mean $\pm\sigma$	mean $\pm\sigma$
OGR-5	11	11	11.0 ± 0.0	10.3 ± 9.7
OGR-6	17	17	17.0 ± 0.0	99.8 ± 65.7
OGR-7	25	25	25.0 ± 0.0	213.2 ± 135.2
OGR-8	34	34	34.0 ± 0.0	114.1 ± 113.0
OGR-9	44	44	44.5 ± 0.5	346344.0 ± 307876.9
OGR-10	59	59	59.6 ± 0.7	433369.0 ± 259147.5
OGR-11	74	74	74.1 ± 0.3	436378.1 ± 270748.4
OGR-12	95	96	96.1 ± 0.7	523898.1 ± 221195.5
OGR-13	117	119	119.1 ± 1.3	517139.2 ± 288309.9
OGR-14	141	147	146.9 ± 2.9	336471.1 ± 296678.9
OGR-15	161	179	176.7 ± 5.8	371481.4 ± 296244.4
OGR-16	209	212	211.6 ± 0.9	185518.7 ± 312827.9

pseudo-greedy mapping from genotype to phenotype; secondly, the representation turns out to be orthogonal, and hence standard string-oriented operators for recombination and mutation can be used. Of course, this does not preclude using those *ad hoc* operators that might be defined. In this sense, reinforcement learning techniques are prime candidates for defining these informed operators.

The hybrid EA has been applied to the Golomb ruler problem, with encouraging results: reactive GRASP and plain EA approaches could be beaten. A permutational EA based on the proposal of [17] was also outperformed. In this sense, it must be noted that results similar to those of the hybrid EA have been reported for this latter permutational EA when executed in long runs using larger population sizes. In a forthcoming work [23], Tavares *et al.* also propose the utilization of *insertion* and *correction* procedures for improving the performance of the algorithm. We believe that these ideas can be fruitfully combined with the hybrid model presented in this work. Another interesting line for future developments lies in the combination of EAs and constraint-satisfaction techniques (e.g., [24]). Work is in progress in this area.

Table 3. Results (averaged for 30 runs) of the permutational EA and the hybrid EA-GRASP.

	Permutational EA			
	ruler length			evaluations
instance	best	median	mean ±σ	mean ±σ
OGR-5	11	11	11.0 ± 0.0	217.8 ± 192.0
OGR-6	17	17	17.0 ± 0.0	514.8 ± 348.1
OGR-7	25	25	25.0 ± 0.0	1821.6 ± 1809.9
OGR-8	34	34	34.1 ± 0.3	157340.7 ± 141192.3
OGR-9	44	44	44.4 ± 0.5	342441.0 ± 254397.7
OGR-10	55	60	59.2 ± 2.3	348509.7 ± 285652.0
OGR-11	74	75	75.7 ± 1.9	468586.8 ± 257449.5
OGR-12	93	95	94.6 ± 1.5	464557.5 ± 257854.7
OGR-13	113	115	115.1 ± 1.0	296010.0 ± 267214.4
OGR-14	143	148	148.0 ± 3.1	548529.3 ± 328350.1
OGR-15	174	177	177.4 ± 2.7	537718.5 ± 259515.4
OGR-16	207	214	213.1 ± 4.1	283229.1 ± 259644.9
	Hybrid GRASP-EA			
	ruler length			evaluations
instance	best	median	mean ±σ	mean ±σ
OGR-5	11	11	11.0 ± 0.0	9.9 ± 29.7
OGR-6	17	17	17.0 ± 0.0	39.6 ± 65.7
OGR-7	25	25	25.0 ± 0.0	514.8 ± 444.5
OGR-8	34	34	34.0 ± 0.0	2465.1 ± 682.2
OGR-9	44	44	44.2 ± 0.4	202761.9 ± 213130.8
OGR-10	55	55	55.4 ± 1.2	245361.6 ± 206552.0
OGR-11	74	74	74.0 ± 0.0	13394.7 ± 10110.6
OGR-12	94	95	94.9 ± 0.3	123542.1 ± 93508.0
OGR-13	111	114	114.2 ± 1.5	341708.4 ± 232391.5
OGR-14	135	139	138.6 ± 2.2	461062.8 ± 271222.9
OGR-15	162	166	166.3 ± 2.2	523531.8 ± 273372.0
OGR-16	189	197	196.5 ± 3.4	429877.8 ± 260082.7

Acknowledgements

The authors wish to thank Jorge Tavares for kindly providing copies of [17] and [23]. This work is partially supported by Spanish MCyT and FEDER under contract TIC2002-04498-C05-02.

References

1. Babcock, W.: Intermodulation interference in radio systems. Bell Systems Technical Journal (1953) 63–73
2. Bloom, G., Golomb, S.: Aplications of numbered undirected graphs. Proceedings of the IEEE **65** (1977) 562–570
3. Feeney, B.: Determining optimum and near-optimum golomb rulers using genetic algorithms. Master thesis, Computer Science, University College Cork (2003)
4. Rankin, W.: Optimal golomb rulers: An exhaustive parallel search implementation. Master thesis, Duke University Electrical Engineering Dept., Durham, NC (1993)

5. Shearer, J.: Some new optimum golomb rulers. IEEE Transactions on Information Theory **36** (1990) 183–184
6. Klove, T.: Bounds and construction for difference triangle sets. IEEE Transactions on Information Theory **35** (1989) 879–886
7. Shearer, J.B.: Golomb ruler table. Mathematics Department, IBM Research, `http://www.research.ibm.com/people/s/shearer/grtab.html` (2001)
8. Schneider, W.: Golomb rulers. MATHEWS: The Archive of Recreational Mathematics, `http://www.wschnei.de/number-theory/golomb-rulers.html` (2002)
9. Garry, M., Vanderschel, D., et al.: In search of the optimal 20, 21 & 22 mark golomb rulers. GVANT project, `http://members.aol.com/golomb20/index.html` (1999)
10. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6** (1995) 109–133
11. Mirchandani, P., Francis, R.: Discrete Location Theory. Wiley-Interscience (1990)
12. Soliday, S., Homaifar, A., Lebby, G.: Genetic algorithm approach to the search for golomb rulers. In Eshelman, L., ed.: 6th International Conference on Genetic Algorithms (ICGA'95), Pittsburgh, PA, USA, Morgan Kaufmann (1995) 528–535
13. Houck, C., Joines, J., Kay, M., Wilson, J.: Empirical investigation of the benefits of partial lamarckianism. Evolutionary Computation **5** (1997) 31–60
14. Julstrom, B.: Comparing darwinian, baldwinian, and lamarckian search in a genetic algorithm for the 4-cycle problem. In Brave, S., Wu., A., eds.: Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference, Orlando, FL (1999) 134–138
15. Giraud-Carrier, C.: Unifying learning with evolution through baldwinian evolution and lamarckism: A case study. In Zimmermann, H.J., Tselentis, G., van Someren, M., Dounias, G., eds.: Advances in Computational Intelligence and Learning: Methods and Applications. Kluwer Academic Publishers (2002) 159–168
16. Moscato, P.: Memetic algorithms: A short introduction. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, Maidenhead, Berkshire, England, UK (1999) 219–234
17. Pereira, F., Tavares, J., Costa, E.: Golomb rulers: The advantage of evolution. In Moura-Pires, F., Abreu, S., eds.: Progress in Artificial Intelligence, 11th Portuguese Conference on Artificial Intelligence. Number 2902 in Lecture Notes in Computer Science, Berlin Heidelberg, Springer-Verlag (2003) 29–42
18. Bean, J.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing **6** (1994) 154–160
19. Resende, M., Ribeiro, C.: Greedy randomized adaptive search procedures. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers, Boston MA (2003) 219–249
20. Prais, M., Ribeiro, C.: Parameter variation in GRASP procedures. Investigación Operativa **9** (2000) 1–20
21. Prais, M., Ribeiro, C.: Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. INFORMS Journal on Computing **12** (2000) 164–176
22. Radcliffe, N.: Equivalence class analysis of genetic algorithms. Complex Systems **5** (1991) 183–205
23. Tavares, J., Pereira, F., Costa, E.: Understanding the role of insertion and correction in the evolution of golomb rulers. In: Congress on Evolutionary Computation Conference (CEC2004), Portland, Oregon, IEEE (2004)
24. Galinier, P., Jaumard, B., Morales, R., Pesant, G.: A constraint-based approach to the golomb ruler problem. In: Third International Workshop on Integration of AI and OR Techniques, Kent, UK (2001) 321–324