

A Memetic Approach to Golomb Rulers

Carlos Cotta¹, Iván Dotú², Antonio J. Fernández¹, and Pascal Van Hentenryck³

¹ Dpto. de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Spain

² Dpto. de Ingeniería Informática, Universidad Autónoma de Madrid, Spain

³ Brown University, Box 1910, Providence, RI 02912

Abstract. Finding Golomb rulers is an extremely challenging optimization problem with many practical applications. This problem has been approached by a variety of search methods in recent years. We consider in this work a hybrid evolutionary algorithm that incorporates ideas from greedy randomized adaptive search procedures (GRASP), tabu-based local search methods (TS) and scatter search (SS). In particular, GRASP and TS are embedded into a SS algorithm to serve as initialization and restarting methods for the population and as improvement technique respectively. The resulting memetic algorithm significantly outperforms earlier approaches (including other hybrid EAs, as well as hybridizations of local search and constraint programming), finding optimal rulers where the mentioned techniques failed.

1 Introduction

Golomb Rulers [1, 2] are a class of undirected graphs that, unlike usual rulers, measure more discrete lengths than the number of marks they carry. More formally, a n -mark Golomb ruler is an ordered sequence of n distinct nonnegative integers $\langle m_1, \dots, m_n \rangle$ ($m_i < m_{i+1}$) such that all distances $m_j - m_i$ ($1 \leq i < j \leq n$) are distinct. Each integer m_i corresponds to a mark on the ruler and the *length* of the ruler is the difference $m_n - m_1$. By convention, the first mark m_1 can be placed in position 0, in which case the length is given by m_n .

The particularity of Golomb Rulers that on any given ruler, all differences between pairs of marks are unique makes them really interesting in many practical applications (cf. [3, 4]). It turns out that finding optimal or near-optimal Golomb rulers (a n -mark Golomb ruler is optimal if there exists no n -mark Golomb ruler of smaller length) is an extremely challenging combinatorial problem. To date, the highest number of marks for which the optimal Golomb ruler (OGR) is known is 23 marks⁴ [7, 8]. Finding optimal Golomb rulers has thus become a standard benchmark to evaluate and compare a variety of search techniques. In particular, evolutionary algorithms (EAs), constraint programming (CP), local

⁴ The search for an optimal 19-marks Golomb ruler took approximately 36,200 CPU hours on a Sun Sparc workstation using a very specialized algorithm [5]. Optimal solutions for 20 up to 23 marks were obtained by massive parallelism projects, taking several months for each of those instances [4, 6].

search (LS), and their hybridizations have all been applied to this problem (e.g., [3, 9–13]).

In this paper, we present a hybrid EA designed to find optimal or near-optimal Golomb Rulers. This algorithm makes use of both an indirect approach and a direct approach in different stages of the search. More specifically, the indirect approach is used in the phases of initialization and restarting of the population and takes ideas borrowed from the GRASP-based evolutionary approach published in [9]. The direct approach is considered in the stages of recombination and local improvement; particularly, the local improvement method is based on the tabu search (TS) algorithm described in [14]. Experimental results show that this algorithm succeeds where another evolutionary algorithms did not. OGRs up to 15 marks (included) can now be found. Moreover, the algorithm produces Golomb rulers for 16 marks that are very close to the optimal value (i.e., 1.1% far), thus improving significantly the results previously reported in the EA literature.

2 Related Work

Two main approaches can be essentially considered for tackling the OGR problem with EAs. The first one is the *direct* approach, in which the EA conducts the search in the space \mathcal{S}_G of all possible Golomb rulers. The second one is the *indirect* approach, in which an auxiliary \mathcal{S}_{aux} space is used by the EA. In this latter case, a decoder [15] must be utilized in order to perform the $\mathcal{S}_{aux} \rightarrow \mathcal{S}_G$ mapping. Examples of the former (direct) approach are the works of Soliday *et al.* [13], and Feeney [3]. As to the latter (indirect) approach, we can cite the work by Pereira *et al.* [10] (based on the notion of random-keys [16]), and Cotta and Fernández [9] (based on ideas from GRASP [17]). This latter paper is specifically interesting since generalizations of the core idea presented there have been used in this work. To be precise, the key idea was using a problem-aware procedure (inspired in GRASP) to perform the genotype-to-phenotype mapping. This method ensured the generation of feasible solutions, and was shown to outperform other previous approaches.

Another very relevant proposal has been recently presented by Dotú and Van Hentenryck [14]. They used a hybrid evolutionary algorithm (GROHEA) that incorporated a tabu-search algorithm for mutation. The basic idea was to optimize the length of the rulers indirectly by solving a sequence of feasibility problems (starting from an upper bound l and producing a sequence of rulers of length $l_1 > l_2 > \dots > l_i > \dots$). This algorithm performed very efficiently and was able to find OGRs for up to 14 marks. Notice that this method requires an estimated initial upper bound, something that clearly favored its efficiency. At any rate, GROHEA outperforms earlier approaches and will be used to benchmark our algorithm.

3 Scatter Search for the OGR Problem

Scatter search (SS) is a metaheuristic based on population-based search whose origin can be traced back to the 1970s in the context of combining decision rules and problem constraints [18]. Among the salient features of SS we can cite the absence of biological motivation, and the emphasis put in the use of problem-aware mechanisms, such as specialized recombination procedures, and LS techniques. In a striking example of convergent evolution, these are also distinctive features of memetic algorithms (MAs) [19]. Indeed, SS and MA can be considered two close formulations of a wider underlying optimization paradigm. There is just one remarkable methodological difference between mainstream versions of SS and MAs: unlike other population-based approaches such as MAs, SS relies more on deterministic strategies rather than on randomization. At any rate, this general methodological principle is flexible, and no clear boundary exists between MAs and SS. This is particularly the case in our approach, in which we use a non-deterministic component within our algorithm. For this reason, we use the terms MA and SS interchangeably in the context of this work. In the following we will describe each of the algorithmic components of our algorithm.

3.1 Diversification Generation Method

The diversification generation method serves two purposes in the SS algorithm considered: it is used for generating the initial population from which the reference set will be initially extracted, and it is utilized for refreshing the reference set whenever a restart is needed.

The generation of new solutions is performed by using a randomized procedure that tries to generate diverse solutions. The basic method utilizes the GRASP-decoding techniques introduced in [9]. Solutions are incrementally constructed in the following: in the initial step, only mark $m_1 = 0$ is placed; subsequently, at each step i an ordered list is built using the n first integers l_1, \dots, l_n such that placing a new mark $m_i = m_{i-1} + l_j$, $1 \leq j \leq n$, would result in a feasible Golomb ruler. A random element is drawn from this list, and used to place mark m_i . The process is iterated until all marks have been placed. Notice that this results in a feasible solution.

A variant of this process is used in subsequent invocations to this method for refreshing the population. This variant is related to an additional dynamic constraint that is imposed in the algorithm: in any solution, it must hold that $m_n < L$, where L is the length of the best feasible Golomb ruler found so far. To fulfill this constraint, new solutions are constructed by generating two feasible rules following the procedure described before, and submitting them to the combination method (see Sect. 3.3), which guarantees compliance with the mentioned constraint.

3.2 Local Improvement Method

The improvement method is responsible for enhancing raw solutions produced by the diversification generation method, or by the solution combination method. In

this case, improvement is achieved via the use of a tabu-search algorithm. This TS algorithm works on tentative solutions that may be infeasible, i.e., there may exist some repeated distances between marks. The goal of the algorithm is precisely to turn infeasible rulers into feasible ones, respecting the dynamic constraint $m_n < L$. Whenever this is achieved, a new incumbent solution is obviously found.

To guide the search, the algorithm uses a notion of constraint violations on the distances. The violation $v_\sigma(d)$ of a distance d in a n -mark ruler σ is the number of times distance d appears between two marks in the ruler σ beyond its allowed occurrences, i.e.,

$$v_\sigma(d) = \max(0, \#\{d_{ij} = d \mid 1 \leq i < j \leq n\} - 1) \quad (1)$$

where $d_{ij} = m_j - m_i$. The overall violation $v(\sigma)$ of a n -mark ruler σ is simply the sum of the violations of its distances d , i.e., $v(\sigma) = \sum_{d \in D} v_\sigma(d)$, where $D = \{d_{ij} \mid 1 \leq i < j \leq n\}$.

The moves in the local search consists of changing the value of a single mark. Since marks are ordered, a mark m_x can only take a value in the interval $I_\sigma(x) = [m_{x-1} + 1, m_{x+1} - 1]$. As a consequence, the set of possible moves is $\mathcal{M}(\sigma) = \{(x, p) \mid (1 < x < n) \wedge (p \in I_\sigma(x))\}$. Observe that m_1 is fixed to 0, and m_n is not allowed to grow. To prevent cycling, a tabu list of movements is kept. The list stores triplets $\langle x, p, i \rangle$, where x is a mark, p is a possible position for mark x , and i represents the first iteration where mark x can be assigned to p again. The tabu tenure, i.e., the number of iterations (x, p) stays in the list, is dynamic and randomly generated in the interval $[4, 100]$. For a ruler σ and an iteration k , the set of legal moves is thus defined as

$$\mathcal{M}^+(\sigma, k) = \{(x, p) \in \mathcal{M}(\sigma) \mid \neg \text{tabu}(x, p, k)\}. \quad (2)$$

where $\text{tabu}(x, p, k)$ holds if the assignment $m_x \leftarrow p$ is tabu at iteration k . The tabu status can be overridden whenever an assignment reduces the smallest number of violations found so far. Thus, if σ^* is the ruler with the smallest number of violations found so far, the neighborhood also includes the moves

$$\mathcal{M}^*(\sigma, \sigma^*) = \{(x, p) \in \mathcal{M}(\sigma) \mid v(\sigma[m_x \leftarrow p]) < v(\sigma^*)\} \quad (3)$$

where $\sigma[m_x \leftarrow p]$ denotes the ruler σ where variable m_x is assigned to p . To intensify the search, the current solution is reinitialized to the initial ruler σ_0 (in the actual TS run) whenever no improvement in the number of violations took place for *maxStable* iterations. The algorithm returns the best solution σ^* found. Fig. 1 shows the complete pseudocode of the TS algorithm.

3.3 Solution Combination Method

The combination of solutions is performed using a procedure that bears some resemblance with the GRASP-decoding mentioned in Sect. 3.1. There are some important differences though: firstly, the procedure is fully deterministic; secondly, the solution produced by the method is entirely composed of marks taken

```

1. TS( $\sigma_0$ )
2.    $tabu \leftarrow \{\}$ ;
3.    $\sigma^* \leftarrow \sigma_0$ ;
4.    $k \leftarrow 0$ ;
5.    $s \leftarrow 0$ ;
6.   while  $k \leq maxIt$  &  $v(\sigma) > 0$  do
7.     select  $(x, p) \in \mathcal{M}^+(\sigma, k) \cup \mathcal{M}^*(\sigma, \sigma^*)$ 
8.       minimizing  $v(\sigma[m_x \leftarrow p])$ ;
9.      $\tau \leftarrow \text{RANDOM}([4, 100])$ ;
10.     $tabu \leftarrow tabu \cup \{(x, p, k + \tau)\}$ ;
11.     $\sigma \leftarrow \sigma[m_x \leftarrow p]$ ;
12.    if  $v(\sigma) < v(\sigma^*)$  then
13.       $\sigma^* \leftarrow \sigma$ ;
14.       $s \leftarrow 0$ ;
15.    else if  $s > maxStable$  then
16.       $\sigma \leftarrow \sigma_0$ ;
17.       $s \leftarrow 0$ ;
18.       $tabu \leftarrow \{\}$ ;
19.    else
20.       $s++$ ;
21.       $k++$ ;
22.    return  $\sigma^*$ ;

```

Fig. 1. Pseudocode of the TS algorithm

from either of the parents; finally, the method ensures that the $m_n < L$ constraint is fulfilled.

The combination method begins by building a list \mathcal{L} of all marks x present in either of the parents, such that $x < L$ ⁵. Then, starting from $m_1 = 0$, a new mark x is chosen at each step i such that (i) $m_{i-1} < x$, (ii) there exist $n - i$ marks greater than x in \mathcal{L} , and (iii) a local optimization criterion is optimized. This latter criterion is minimizing $\sum_{j=1}^{i-1} v_\sigma(x - m_j)^2 + (x - m_{i-1})$, where σ is the partial ruler. This expression involves minimizing the number of constraints violated when placing the new mark, as well as the subsequent increase in length of the ruler. The first term is squared to raise its priority in the decision-making.

3.4 Subset Generation and Reference Set Update

This subset generation method creates the groups of solutions that will undergo combination. The combination method used is in principle generalizable to an arbitrary number of parents, but we have considered the standard two-parent

⁵ It might happen that the number of such marks were not enough to build a new ruler. In that case, a plain solution with length ∞ (that is, the worst possible value) is returned.

recombination. Hence the subset generation method has to form pairs of solutions. This is done exhaustively, producing all possible pairs. It must be noted that since the combination method utilized is deterministic, it does not make sense to combine again pairs of solutions that were already coupled before. The algorithm keeps track of this fact to avoid repeating computations.

As to the reference set update method, it must produce the reference set for the next step by using the current reference set and the newly produced offspring (or by using the initial population generated by diversification at the beginning of the run or after a restart). Several strategies are possible here. Quality is an obvious criterion to determine whether a solution can gain membership to the reference set: if a new solution is better than the worst existing solution, the latter is replaced by the former. In the OGR, we consider that a solution x is better than a solution y if the former violates less constraints, or violates the same number of constraints but has a lower length. It is also possible to gain membership of the reference set via diversity. To do so, a subset of *diverse* solutions (i.e., *distant* solutions to the remaining high-quality solutions in the set – an appropriate definition of a distance measure is needed for this purpose) is kept in the reference set, and updated whenever a new solution improves the diversity criterion.

If at a certain iteration of the algorithm no update of the reference set takes place, the current population is considered stagnated, and the restart method is invoked⁶. This method works as follows: let μ be the size of the reference set; the best solution in the reference set is preserved, $\lambda = \mu(\mu - 1)/2$ solutions are generated using the diversification generation method and the improvement method, and the best $\mu - 1$ out of these λ solutions are picked and inserted in the reference set.

4 Experimental Results

To evaluate our memetic approach, a set of experiments for problem sizes ranging from 10 marks up to 16 marks has been realized. In all the experiments, the maximum number of iterations for the tabu search was set to 10,000, the size of the population and reference set was 190 and 20 respectively, and the arity of the combination method was 2. The reference set is only updated on the basis of the quality criterion. One of the key points in the experimentation has been analyzing the influence of the local search strategy with respect to the population-based component. To this end, we have experimented with partial Lamarckism [20], that is, applying the local improvement method just on a fraction of the members of the population. To be precise, we have considered a probability p_{ts} for applying LS to each solution. The values $p_{ts} \in \{0.1, 0.2, 0.4, 0.6, 0.81.0\}$ have been considered. All algorithms were run 20 times until an optimal solution was

⁶ Notice that the TS method used for local improvement is not deterministic. Thus, it might be possible that further applications of TS on the stagnated population resulted in an improvement. However, due to the computational cost of this process, it is advisable to simply restart.

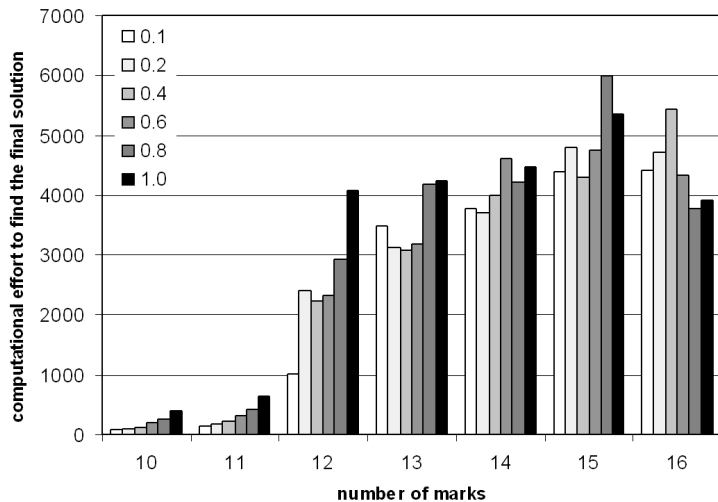
		10	11	12	13	14	15	16
HGRASP	Best	N/A	2.8	10.6	4.7	6.3	7.3	6.8
	Median	N/A	2.8	11.8	7.5	9.4	11.9	11.3
GROHEA	Best	N/A	0	0	0	3.1	4.6	5.6
	Median	N/A	<u>0</u>	7.1	5.6	7.1	8.6	10.2
MA1.0	Best	0	0	0	0	1.6	0	4.0
	Median	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	2.4	4.0	6.2
MA0.8	Best	0	0	0	0	0.8	1.3	2.3
	Median	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1.6</u>	<u>3.3</u>	<u>5.6</u>
MA0.6	Best	0	0	0	0	0.8	0	2.8
	Median	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1.6</u>	4.0	6.2
MA0.4	Best	0	0	0	0	0	1.3	1.1
	Median	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1.6</u>	4.0	<u>5.6</u>
MA0.2	Best	0	0	0	0	0	0.7	3.4
	Median	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1.6</u>	4.0	6.2
MA0.1	Best	0	0	0	0	0	0.7	3.4
	Median	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1.6</u>	<u>3.3</u>	<u>5.6</u>

Table 1. Relative distances to optimum for different probabilities of the MA and the algorithms GROHEA and HGRASP. Globally best results (resp. globally best median results) for each instance size are shown in boldface (resp. underlined). Results of HGRASP and GROHEA are not available for 10 marks.

found, or a limit in the whole number of evaluations was exceeded. This number of evaluations was set so as to allow a fixed average number e of LS invocations ($e = 10,000$ TS runs). Thus, the number of evaluations was limited in each of the instances to e/p_{ts} . This is a fair measure since the computational cost is dominated by the number of TS invocations.

Table 1 reports the experimental results for the different instances considered. Row $MAxx$ corresponds to the execution of the MA with a local improvement rate of $p_{ts} = xx$. The table reports the relative distance (percentage) to the known optimum for the best and median solutions obtained. The table also shows the results obtained by the algorithms described in [9] (i.e., HGRASP) and [14] (GROHEA). Algorithm HGRASP is grounded on the evolutionary use of the GRASP-based solution generation method used in the basic diversification method of our algorithm. As to algorithm GROHEA, it provides the best results reported in the literature for this problem via a population-based approach, and therefore it is the benchmark reference for our algorithm. Specifically for this latter algorithm, as reported in [14], the maximum number of iterations for the tabu search was also 10,000, the size of the population was 50, and the probabilities p_m and p_X were both set to 0.6. Both algorithms (GROHEA and HGRASP) were run 30 times for each ruler.

The results are particularly impressive. Firstly, observe that our memetic algorithm systematically find optimal rulers for up to 13 marks. GROHEA is also capable of eventually finding some optimal solutions for these instance sizes, but notice that the median values are drastically improved in the MA. In fact,



	0.1	0.2	0.4	0.6	0.8	1.0
0.1	•					
0.2	+++--					
0.4	+++--	•				
0.6	+++--		•			
0.8	+++--	+++--	+++--	•		
1.0	+++--	+++--	+++--	+++--	•	

Fig. 2. (Top) Computational effort (measured in number of TS invocations) to find the best solution. (Bottom) Statistical comparison of the computation effort. In each cell, the results ('+'=significant, '-'=non-significant) correspond from left to right to instance sizes from 10 up to 16.

the median values obtained by the MA for these instances correspond exactly to their optimal solutions. Comparatively, the results are even better in larger OGR instances: our MA can find optimal ORGs even for 14 and 15 marks, and computes high-quality near-optimal solutions for 16 (i.e., 1.1% from the optimum). These results clearly outperform GROHEA; indeed, the latter cannot provide optimal values for instance sizes larger than 14 marks. Moreover, all *MAxx* significantly improve the median values obtained by GROHEA on the larger instances of the problem. These results clearly indicate the potential of hybrid EAs for finding optimal and near-optimal rulers.

We have also conducted statistical tests to ascertain whether there are significant performance differences between the different LS application rates. This has been done using a non-parametric Wilcoxon ranksum test (results are not normally distributed). Except in three head-to-head comparisons for 14 marks ($p_{ts} = 1.0$ vs $p_{ts} = 0.8$ and $p_{ts} = 0.1$, and $p_{ts} = 0.4$ vs $p_{ts} = 0.1$), there is no statistically significant difference (at the standard 0.05 level) in any instance size for the different values of p_{ts} . While this is consistent with the fact that the average number of TS invocations is constant, it raises the issue of whether the associated computational cost is the same or not. The answer to this question

can be seen in Fig. 2. As expected, the computational cost increases with the size of the problem. Quite interestingly, the average cost decreases for 16 marks. This behavior owes to the higher difficulty of the problem for this latter size: the algorithm quickly reaches a near-optimal value (a remarkable result), and then stagnates (longer runs would be required to improve the solutions from that point on). The table at the bottom of Fig. 2 shows the outcome of the statistical comparison between the computational cost of the *MAxx* for a given instance size. As it can be seen, the differences are almost always significant for the lower range of sizes, and progressively become non-significant as the size increases. For 16 marks, there is just one case of statistically significant difference of computational cost ($p_{ts} = 0.4$ vs $p_{ts} = 0.8$). Since the small values of p_{ts} imply a lower computational cost for instance sizes in the low range, and there is no significant difference in either quality or computational cost with respect to higher values of p_{ts} in the larger instances, it seems that values $p_{ts} \in \{0.1, 0.2\}$ are advisable.

5 Conclusions

We have presented a memetic approach for finding near-optimal Golomb rulers at an acceptable computational cost. The MA combines, in different stages of the algorithm, a GRASP-like procedure (for diversification and recombination) and tabu search (for local improvement) within the general template of scatter search. The results of the MA have been particularly good, clearly outperforming other state-of-the-art evolutionary approaches for this problem. One of the aspects on which our analysis has been focused is the influence of the LS component. We have shown that lower rates of Lamarckism achieve the best tradeoff between computational cost and solution quality.

We are currently exploring alternatives for some of the operators used in our algorithm. Preliminary experiments with multi-tier reference sets –i.e., including a diversity section– do not indicate significant performance changes. A deeper analysis is nevertheless required here. In particular, it is essential that the particular distance measure used to characterize diversity correlate well with the topology of the search landscape induced by the reproductive operators. Related to this issue, we plan to test alternative recombination methods based on exhaustive techniques used in constraint programming. Defining appropriate distance measures in this context (and indeed, checking their usefulness in practice) will be the subsequent step.

Acknowledgements. This work was partially supported by Spanish MCyT under contracts TIN2004-7943-C04-01 and TIN2005-08818-C04-01.

References

1. Babcock, W.: Intermodulation interference in radio systems. *Bell Systems Technical Journal* (1953) 63–73

2. Bloom, G., Golomb, S.: Applications 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. Dollas, A., Rankin, W.T., McCracken, D.: A new algorithm for Golomb ruler derivation and proof of the 19 mark ruler. *IEEE Transactions on Information Theory* **44** (1998) 379–382
6. 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)
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. Cotta, C., Fernández, A.: A hybrid GRASP - evolutionary algorithm approach to Golomb ruler search. In Yao, X., et al., eds.: *Parallel Problem Solving From Nature VIII*. Number 3242 in *Lecture Notes in Computer Science*, Birmingham, UK, Springer (2004) 481–490
10. 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
11. Prestwich, S.: Trading completeness for scalability: Hybrid search for cliques and rulers. In: *Third International Workshop on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CPAIOR 2001*, Ashford, Kent, England (2001)
12. Smith, B., Stergiou, K., Walsh, T.: Modelling the golomb ruler problem (1999)
13. 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
14. Dotú, I., Hentenryck, P.V.: A simple hybrid evolutionary algorithm for finding golomb rulers. In et. al., D.C., ed.: *Congress on Evolutionary Computation Conference (CEC2005)*. Volume 3., Edinburgh, Scotland, IEEE (2005) 2018–2023
15. Koziel, S., Michalewicz, Z.: A decoder-based evolutionary algorithm for constrained parameter optimization problems. In Bäck, T., Eiben, A., Schoenauer, M., Schwefel, H.P., eds.: *Parallel Problem Solving from Nature V*. Volume 1498 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg (1998) 231–240
16. Bean, J.: Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing* **6** (1994) 154–160
17. 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
18. Laguna, M., Martí, R.: *Scatter Search. Methodology and Implementations in C*. Kluwer Academic Publishers, Boston MA (2003)
19. Moscato, P., Cotta, C.: A gentle introduction to memetic algorithms. In Glover, F., Kochenberger, G., eds.: *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston MA (2003) 105–144
20. Houck, C., Joines, J., Kay, M., Wilson, J.: Empirical investigation of the benefits of partial lamarckianism. *Evolutionary Computation* **5** (1997) 31–60