Constraints, , 1–30 ()
© Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Local Search-Based Hybrid Algorithms for Finding Golomb Rulers

CARLOS COTTA ccottap@lcc.uma.es
Dept. Lenguajes y Ciencias de la Computación, E.T.S.I. Informática, Universidad de Málaga, Campus de Teatinos, 29071 – Málaga, Spain

IVÁN DOTÚ ivan.dotu@uam.es
Dept. Ingeniería Informática, Universidad Autónoma de Madrid, Spain

ANTONIO J. FERNÁNDEZ afdez@lcc.uma.es
Dept. Lenguajes y Ciencias de la Computación, E.T.S.I. Informática, Universidad de Málaga, Campus de Teatinos, 29071 – Málaga, Spain

PASCAL VAN HENTENRYCK pvh@cs.brown.edu
Brown University, Box 1910, Providence, RI 02912

Abstract. The Golomb Ruler Problem is a very hard combinatorial optimization problem that has been tackled with many different approaches, such as Constraint Programming (CP), Local Search (LS), Evolutionary Algorithms (EAs), and hybrid LS and CP, among others.

This paper describes several local search-based hybrid algorithms to find optimal or near-optimal Golomb rulers. These algorithms are based on both stochastic methods and systematic techniques. More specifically, the algorithms combine ideas from greedy randomized adaptive search procedures (GRASP), scatter search (SS), tabu search (TS), clustering techniques, and constraint programming (CP). Each new algorithm is, in essence, born from the conclusions extracted after the observation of the previous one. With these algorithms we are capable of solving large rulers with a reasonable efficiency. In particular, we can now find optimal Golomb rulers for up to 16 marks.

In addition, the paper also provides an empirical study of the fitness landscape of the problem with the aim of shedding some light about the question of what makes the Golomb ruler problem hard for certain classes of algorithm.

Keywords: Constraint Programming, Local Search, Tabu Search, Evolutionary Algorithms, Golomb Rulers, Clustering.

1. Introduction

The concept of Golomb rulers was first introduced by W.C. Babcock in 1953 [2], and further described by S.W. Golomb [6]. 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 all differences between pairs of marks are unique. Precisely, this feature makes Golomb rulers really interesting in many applications in the real world, mainly in engineering. For instance, in the field of communications when setting up an interferometer for radio astronomy, placing the antennas on the marks of a Golomb ruler maximizes the recovery of information about the phases of the signal received [4]. Golomb rulers have been used for the design of self-orthogonal codes in the area of coding theory (for error detection and correction) [30, 49]. Moreover, Golomb rulers have been applied to many more interesting applications such as carrier frequency assignment [17], radio communication [2], X-ray crystallography [6], and pulse phase modulation [51], among others (see a description of a number of practical applications in [14, 15, 48]). Needless to say, this feature also introduces numerous constraints that hinder the search of short feasible rulers, let alone *optimal* Golomb rulers (an *Optimal Golomb Ruler* – OGR – is defined as the shortest Golomb ruler for a given 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 [54]. This has been a major limitation as each new ruler to be discovered is, by necessity, larger than its predecessor. In any case, the search space is bounded and, therefore, solvable [30]. To date, the largest number of marks for which an optimal Golomb ruler has been found is 24 marks. Finding optimal Golomb rulers with a high number of marks is, computationally speaking, very costly. For instance, 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 [14]. Moreover, optimal solutions for 20 up to 24 marks were obtained by massive parallelism projects, taking several months -even years for 24 marks - for each of those instances [48, 24]. [53, 55]. 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) and local search (LS). Being such an extremely difficult combinatorial task, the Golomb ruler problem represents an ideal scenario for deploying the arsenal of search algorithms.

The main contribution of this paper is the introduction of several hybrid approaches based on EAs, LS, and CP to solve the OGR problem. Combining ideas from greedy randomized adaptive search procedures (GRASP) [16], scatter search (SS) [23, 32], tabu search (TS) [21, 22], clustering [27], and CP, we are capable of solving the OGR problem for up to 16 marks, a notorious improvement with regard to previous approaches reported in the literature. We present successive algorithmic models that integrate these techniques, and show that they succeed in problem instances where other approaches could not. We also study the search space of the OGR problem from a general standpoint, and provide some hints on what makes the problem hard (or more precisely, on why some representations of the problem can be inappropriate for certain search algorithms). Indeed, one of the key aspects of our approaches is the eclectic combination of direct and indirect formulations of solutions in different stages of the search. Before providing more details on this, the following section will describe the notation used in the rest of the paper, and will survey previous related work on the OGR problem.

2. Background

The OGR problem can be classified as a fixed-size subset selection problem, such as e.g., the p-median problem [40]. It exhibits some very distinctive features though.

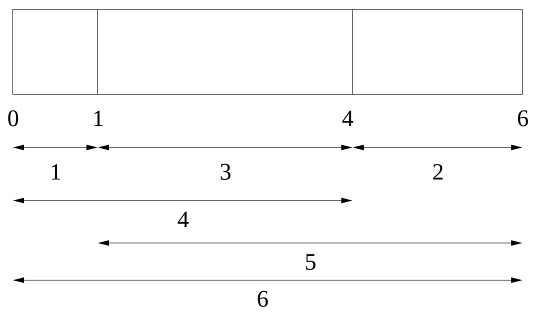


Figure 1: A Golomb Ruler with 4 marks

A brief overview of the problem, and how it has been tackled in the literature is provided below.

2.1. Golomb Rulers: definition

A Golomb ruler consists of ordered series of non-negative integer numbers. These numbers are referred to as *marks*, and corresponds to positions on a linear scale. The difference between the values of any two marks is called *the distance* between two marks, and this distance is unique for that ruler. The difference between the first and last mark is referred to as the *length* of the Golomb ruler, and corresponds to the largest distance for the Golomb ruler. By convention, it is assumed that the first mark of the ruler is placed at position 0. For a specific number of marks, there may exist multiple different Golomb rulers of the same length, and each Golomb ruler of n-marks has an equivalent ruler, called the *mirror image*. In the mirror image, the n-th mark of the original ruler will be at the extreme left of the image. The new position of each mark, originally at, say i, will now be at $n - i$. The properties of mirror images are that they have identical first-order differences with the result that they measure an identical set of distances. This is important because no pair of rulers with greater than six marks have been found to have the property unless they are mirror images [48].

In other words, an *n marks Golomb ruler* is a sequence of n distinct non-negative integers, called *marks*, $a_1 < ... < a_n$, such that all the differences $a_i - a_j$ $(i > j)$ are mutually 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 marks Golomb ruler having smaller length, and
- the ruler is canonically “smaller” with respect to its equivalent. This means that the distance between the first two marks in the ruler is less than the corresponding distance in the equivalent ruler.

Optimality is not unique in Golomb rulers and there may be more than one optimal Golomb ruler with a specific number of marks.

2.2. Representational issues

In essence, two main approaches can be distinguished for tackling this problem. The first one is the *direct* approach in which the algorithm conducts the search in the space $\mathbb{G}_n$ of all the n marks Golomb rulers (i.e., the solution set for n marks). In this approach, Golomb rulers are typically represented by the values of the marks on the ruler, i.e., in an n marks Golomb ruler, $a_i = k$ $(1 \leqslant i \leqslant n)$ means that k is the value of the mark in position i. Fig.1 shows an OGR with 4 marks (observe that all distances between any two marks are different) and the sequence $\langle 0, 1, 4, 6\rangle$ would represent *directly* the ruler. Its mirror image would be then represented by the sequence $\langle 0, 2, 5, 6\rangle$.

However, this representation turns out to be inappropriate for several approaches, such as for EAs (for example, it is problematic with respect to developing good crossover operators [56]). For that reason, an alternative approach is the so-called *indirect* approach, in which an auxiliary $\mathcal{S}_{\text{aux}}$ space is used by the EA in order to perform the $\mathcal{S}_{\text{aux}} \longrightarrow \mathbb{G}_n$ mapping. For instance, an alternative representation consists of representing the Golomb ruler by the lengths of its segments, i.e., the distances between consecutive marks. Therefore, a Golomb Ruler can be represented with $n-1$ distances specifying the lengths of the $n-1$ segments that compose it. Following the previous example, the sequence $\langle 1, 3, 2\rangle$ would encode the ruler in Fig.1.

2.3. Finding OGRs

The OGR problem has been solved using very different techniques. The evolutionary techniques found in the literature to obtain OGRs are described in Sect. 2.4 since they are closer to our hybrid algorithms and we believe they should be reviewed separately for that reason. Here, we provide a brief overview of some of the most popular non-evolutionary techniques used for this problem.

Firstly, it is worth mentioning some classical algorithms used to generate and verify OGRs such as the *Scientific American* algorithm [12], the *Token Passing* algorithm (created by Professor Dollas at Duke University) and the *Shift* algorithm [37], all of them compared and described in [48]. In general, both non-systematic and systematic methods have been applied to find OGRs. Regarding the former, we can cite for example the use of geometry tools (e.g., projective plane construction and affine plane construction). With these approaches, one can compute very good approximate values (i.e., near-optimal Golomb rulers whose lengths are within several units of the corresponding optimal Golomb rulers) for OGR with up to 158 marks [55]. As to systematic (exact) methods, we can mention the method proposed by Shearer to compute OGRs up to 16 marks [54]; basically this method was based on the utilization of branch-and-bound algorithms combined with a depth first search strategy (i.e., backtracking algorithms), making use of upper-bounds set equal to the minimum length in the experiments. Evidently this avoided the divergence of the results and influenced positively the performance of the algorithm. This approach has been also followed in massive parallelism initiatives such as the

Table 1: Experimental results of CLS for rulers from 5 to 16 marks. Entries indicate the relative error with respect to the optimum of the best solution found

	number of marks											
	5	6	7	8	9	10	11	12	13	14	15	16
Relative errors	0.0	0.0	0.0	0.0	0.0	0.0	0.0	11.8	6.6	9.4	10.6	12.99

OGR project mentioned before. This project has been able to find the OGRs with a number of marks between 20 and 24, although it took several months (even years) to find optimum for each of those instances [14, 24, 42, 48].

Constraint programming techniques have also been used, although with limited success. For example, Smith and Walsh [57] obtained interesting results in terms of nodes in the branching schema. However, computation times are far from the results obtained by previous approaches. More recently, Galinier *et al.* [19] proposed a combination of constraint programming and sophisticated lower bounds for finding OGRs. They showed that both the shape of the search tree (i.e., basically the number of nodes) and the computation time depended on the different ways in which the bound might be used.

A novel hybrid local search and constraint programming approach is presented in [45]. The author develops a hybrid algorithm named Constrained Local Search (CLS), which introduces a noise parameter b that indicates the number of variables to unassign when a dead-end occurs. This parameter can take values $b \geq 1$ and the heuristic to choose the variables to unassign is partly random and makes no attempt to maintain completeness. For the Golomb Ruler Problem, the model is based on the *ternary and binary constraint* model described in [57]. Table 1 shows the experimental results for this approach.

2.4. Evolutionary Approaches to the OGR

In this section we will focus on the evolutionary approaches to solve OGRs considered so far in the literature. The direct and indirect approaches, as presented in Sect. 2.2, are discussed below.

2.4.1. Direct Approaches. In 1995, Soliday, Homaifar and Lebby [56] 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 $n-1$ were selected and position 1 (initially containing the value 1) was randomly swapped with some of those selected positions. Soliday *et al.* also prevented mirror

Table 2: Results obtained by Soliday *et al.*'s and Feeney's approaches. Entries indicate the relative error with respect to the optimum of the best solution found.

	number of marks											
	5	6	7	8	9	10	11	12	13	14	15	16
Soliday *et al.*	0	0	0	2.9	0	12.7	9.7	21.2	17.0	32.3	36.4	36.5
Feeney	0	0	0	0	6.8	12.7	11.1	18.8	15.1	15.0	18.9	20.3

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 a 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 [6]. A special crossover operator was designed to guarantee that descendants are valid permutations and that length 1 was also retained.

Later, Feeney studied the effect of hybridizing genetic algorithms with local improvement techniques to solve Golomb rulers [15]. Namely, he 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. [18, 25, 29] for more information on Lamarckian and Baldwinian learning). It is known that, combined with EAs, local search techniques often reduce drastically the number of generations to find a near-optimal solution (see e.g., [41]). 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 usually depend greatly on the problem and even on the instance 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 2 shows the results produced by Soliday *et al.*'s and Feeney's approaches. With respect to the different approaches studied by Feeney, we have selected the best result obtained in [15], that corresponds to the execution of a genetic algorithm (with high mutation and no crossover) without local search. Here, the maximum number of generations allowed was 250; the program was executed three times and the ruler whose length was closest to the average obtained in the three executions was accepted as the most typical result (unless two had the same length in which case the most typical result was their length). 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 9.7% and 36.5%. On its turn, Feeney's approach behaves better and maintains this error between 6.8% and 20.3% for the same rulers, which means a significant improvement.

Feeney's hybrid approach was found to be successful in generating near-optimal rulers and in generating optimal Golomb rulers of a short length. Observe also that the error remains in the short range [15.0%..20.3%] for rulers with marks from 12 to 16. This clearly indicates a stabilization of the error. However, as it is reported in [15], performance was really poor, as it was expected, mainly due to the local search.

2.4.2. Indirect Approaches. Recently, Pereira *et al.* [46] have presented a new EA approach to find OGRs. This new approach uses the concept of random keys [5] to codify the information contained in each chromosome. As in the Soliday *et al.*'s approach, any candidate solution for an n marks OGR is represented with the length of each of its $n-1$ segments. In fact, a chromosome is composed by a permutation of λ distinct values (where λ is the maximum segment length), and the encoding of the permutation is done with random keys (RK). The basic idea consists of generating a sequence of n random numbers (i.e., the keys) sampled from the interval $[0,1]$; the positions in the sequence are then sorted in a decreasing order, according to the key they store. This results in a permutation of the n indices. The codification is based on a more advanced principle: 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 [56], 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

Our first approach to tackle the Golomb ruler problem is an indirect approach based on incorporating ideas from greedy randomized adaptive search procedures (GRASP) to an evolutionary algorithm [9, 16]. It is thus necessary discussing firstly the deployment of this latter metaheuristic on the OGR problem.

3.1. Basic GRASP for the OGR Problem

GRASP can be viewed as repetitive sampling techniques [52]. Each iteration, 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 solutions incrementally 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 is shown in Fig. 2.

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

Figure 2: Pseudocode of the greedy construction process

One of the key steps in this pseudocode is the selection of an attribute from *RCL* (i.e., the ranked candidate list). 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 ith 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 ith 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 [43], 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 [44]. 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 [44], 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, so that it is possible to use different values of the parameter not just in each application of the construction phase, but in each internal step of the construction algorithm. Here 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 (we will denote this approach as HEAGRASP after hybrid EA-GRASP). 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 ith iteration of the construction algorithm. Two practical considerations 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_ith best attribute value should be selected in the ith 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 [47], i.e., any sequence represents a feasible solution, and hence, standard operators for crossover and mutation can be used to manipulate them.

3.3. *Experimental Results*

The experiments have been performed using four different algorithms: plain GRASP, reactive GRASP, a permutational EA following [46], and HEAGRASP. 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

Table 3: Relative distances to optimum for plain GRASP, reactive GRASP, the permutational EA (permEA), and heagrasp. Globally best results (resp. globally best median results) for each instance size are shown in boldface (resp. underlined).

	plain GRASP		reactive GRASP		permEA		heagrasp	
instance	best	median	best	median	best	median	best	median
OGR-5	**0**	0	**0**	0	**0**	0	**0**	0
OGR-6	**0**	0	**0**	0	**0**	0	**0**	0
OGR-7	**0**	0	**0**	0	**0**	0	**0**	0
OGR-8	**0**	0	**0**	0	**0**	0	**0**	0
OGR-9	**0**	0	**0**	0	**0**	0	**0**	0
OGR-10	9.1	12.7	7.3	7.3	**0**	9.1	**0**	0
OGR-11	8.3	12.5	2.8	2.8	**2.8**	4.2	**2.8**	2.8
OGR-12	21.2	23.5	11.8	12.9	**9.4**	11.8	10.6	11.8
OGR-13	19.8	21.7	10.4	12.3	6.6	8.5	**4.7**	7.5
OGR-14	16.5	28.3	11.0	15.7	12.6	16.5	**6.3**	9.4
OGR-15	26.5	32.5	6.6	18.5	15.2	17.2	**7.3**	9.9
OGR-16	27.7	36.7	18.1	19.8	16.9	20.9	**6.8**	11.3

hybrid EA, each gene can take values $r_i \in [1..n]$, 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 [46]. Random keys are here directly substituted by permutations, being partially-mapped crossover (PMX) [20] used for recombination, and the swap operator [39] 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 3. 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. 2.3.

Focusing on the EAs, notice firstly that the permutational EA provides comparable results to those of reactive GRASP. heagrasp 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 [31], since the results are not normally distributed– indicates that the differences are significant for 10, 11, 14, 15, and 16 marks.

The heagrasp algorithm also performed efficiently regarding computation time as the average time of the heagrasp algorithm for solving the OGR problem instances of 11, 12, 13, 14, 15 and 16 marks was 1.5, 2.4, 3.6, 5.3, 7.6 and 11.3 minutes respectively.

4. Fitness Landscapes for the Golomb Ruler Problem

The notion of fitness landscapes was firstly introduced in [58] to model the dynamics of evolutionary adaptation in nature. The fitness landscape analysis of a problem can help to identify its structure in order to improve the performance of search algorithms (e.g., to predict the behavior of a heuristic search algorithm, or to exploit some of its specific properties). For this reason, this kind of analysis has become a valuable tool for evolutionary-computation researchers.

This section tries to shed some light on the question of what makes the Golomb ruler problem hard for certain types of search algorithm. The OGR problem is also a problem for which several representations had been tried, but that lacked an analysis of the combinatorial properties of the associated fitness landscapes. Here, we will analyze the fitness landscapes resulting from the two problem representations described before, the classical direct encoding of rulers, and the use of a GRASP-based decoder. We will assume below that n is the number of marks for a specific Golomb ruler, and that $a = \langle a_1, \ldots, a_n \rangle$ and $b = \langle b_1, \ldots, b_n \rangle$ are arbitrary solutions from $\mathbb{G}_n$. Analogously, $r = \langle r_1, \cdots, r_{n-2} \rangle$ and $r' = \langle r'_1, \cdots, r'_{n-2} \rangle$ are arbitrary vectors from $\mathbb{N}^{n-2}$, representing the vector of indices for selecting segment lengths. We denote by ψ the bijective function performing the genotype-to-phenotype mapping $\mathbb{N}^{n-2} \to \mathbb{G}_n$.

4.1. Distance Measures and Neighborhood Structure

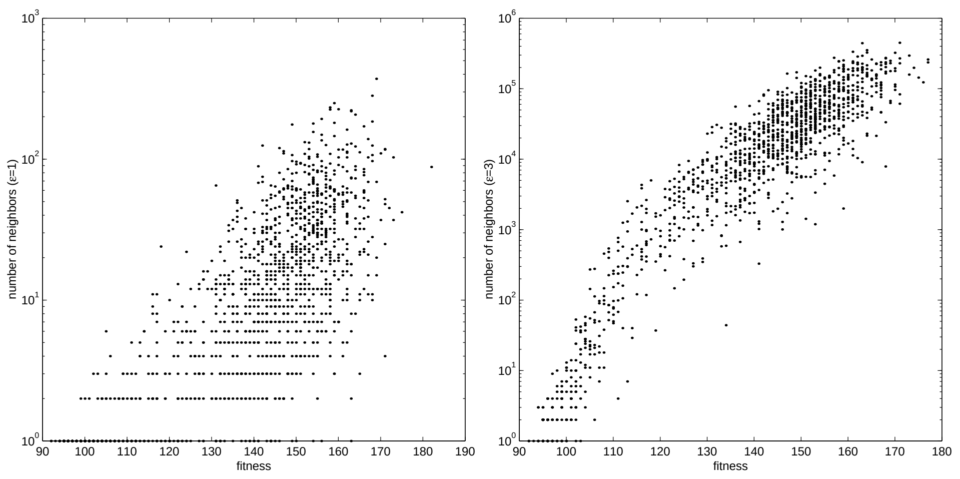
We define a fitness landscape for the OGR as a triple $\langle \mathbb{G}_n, f, d \rangle_n$ where, as already mentioned, $\mathbb{G}_n$ is the set of all the n marks Golomb rulers (i.e., the solution set), f is a fitness function that attaches a fitness value to each of the points in $\mathbb{G}_n$ (i.e., $f(a)$ is equal to a_n, the length of a), and $d : \mathbb{G}_n \times \mathbb{G}_n \to \mathbb{N}$ is a function that measures a distance between any two points in $\mathbb{G}_n$. We have defined one distance function for each of the Golomb ruler representations already commented. Specifically for the direct formulation (i.e., that based on lists of marks) we have defined the distance function d as follows:

$$d(a, b) = \max\{\mid b_i - a_i \mid, 1 \leq i \leq n\} \ . \tag{1}$$

In other words, $d(a, b)$ returns the maximum difference between any two corresponding marks in a and b. Also, for our indirect formulation (i.e., the GRASP-based formulation) we have defined the distance function d as the L_1 norm (the Manhattan distance) on the vector of indices, i.e.,

$$d(a, b) = d(\psi(r), \psi(r')) = \sum_{i=1}^{n-2} \mid r_i - r'_i \mid \ . \tag{2}$$

A first issue to be analyzed regards the neighborhood structure induced by these distance measures. More precisely, consider the number of solutions reachable from a certain point in the search space, by a search algorithm capable of making jumps of a given distance. In the direct formulation, this number of solutions turns out to



(a) (b)

Figure 3: Number of neighbors for different values of the local radius ϵ in a 12 marks Golomb ruler problem. (a) $\epsilon = 1$, (b) $\epsilon = 3$. Notice the log-scale in the Y-axis.

be variable for each point of $\mathbb{G}_n$, as shown in Fig. 3. We have implemented and used a logic-programming based constraint solver to solve the Golomb ruler constraint satisfaction problem for an arbitrary number of marks. Our solver, implemented in GNU Prolog [11], is based on the model proposed in [3]. In particular, the solver generates a list of all possible distances between any pair of marks i, j ($i < j$ and $i, j \in [1..n]$) in the ruler and then imposes a global constraint *all-different* on this list instead of imposing the set of binary inequalities between any two marks i, j. The efficiency is further improved by adding some redundant constraints leading to an improvement of the domain pruning. This solver calculates the number of possible neighbors that are located within a given distance ϵ (called the *local radius*) of certain solution a (i.e., it obtains the cardinality of the set $\{\langle c_1, \ldots, c_n\rangle \in \mathbb{G}_n \mid a_i - \epsilon \leq c_i \leq a_i + \epsilon, 1 \leq i \leq n\}$). The solver is then applied to a large sample of solutions covering a wide range of fitness values.

The outcome of this experiment indicates that the connectivity of the fitness landscape increases with worse fitness values. Furthermore, this effect is stronger as we increase the neighborhood radius (see Fig.3). This kind of irregularity is detrimental for search algorithm navigating this landscape [8], since the neighborhood structure tends to guide the search towards low-fitness regions. This means that a search algorithm on this landscape would have to be continuously fighting against this drifting force. On the contrary, notice that the fitness landscape of the indirect formulation is perfectly regular, since its topology is isomorphic to $\mathbb{N}^{n-2}$. In principle, this regularity makes this landscape more navigable since no underlying drift effect exists.

4.2. Fitness-Distance Correlation

Fitness-distance correlation (FDC) [28] is one of the most widely used measures for assessing the structure of the landscape. It also constitutes a very informative measure to evaluate the problem difficulty for evolutionary algorithms [26]. FDC allows quantifying the correlation between fitness values, and the distance to the nearest optimum in the search space. Landscapes with a high FDC typically exhibit a *big valley structure* [50] (this is not always the case though [28, 7]).

It is typically assumed that low FDC is associated with problem difficulty for local search. Nevertheless, the interplay of this property with other landscape features is not yet well understood. Indeed, it will be later shown how landscape ruggedness and neighborhood irregularity can counteract high FDC values. Focusing on the problem under consideration, the optimum value opt_n for n marks Golomb rulers is known (up to $n = 24$, enough for our analysis). We can then obtain a sample of m locally-optimal solutions $A = \{a_1, \ldots, a_m\} \subset \mathbb{G}_n$ and easily calculate the sets $F = \{f_i \mid f_i = f(a_i), 1 \leq i \leq m, a_i \in A\}$ and $D = \{d_i \mid d_i = d(a_i, opt_n), 1 \leq i \leq m, a_i \in A\}$. Then we can compute the correlation coefficient as $FDC = C_{FD}/(\sigma_F \sigma_D)$, where

$$C_{FD} = \frac{1}{m} \sum_{i=1}^{m} (f_i - \overline{f})(d_i - \overline{d}) \tag{3}$$

is the covariance of F and D, and $\sigma_F, \sigma_D, \overline{f}$ and $\overline{d}$ are, respectively, the standard deviations and means of F and D. Observe that this definition depends on the definition of the distance function, and as shown in Sect. 4.1, we consider two different definitions for the two problem representations. In all cases, locally optimal solutions are computed by using hill climbing from a fixed sample of seed feasible solutions.

The results indicate a high correlation for the direct formulation, specially for low values of the local radius ϵ. This can be explained by the fact that the fitness of a solution is actually the value of the last mark, and this value will not change above the given ϵ within the neighborhood. FDC starts to degrade for increasing values of this local radius. To be precise, FDC values for $\epsilon = 1$ up to $\epsilon = 4$ are 0.9803, 0.9453, 0.8769, and 0.8221 respectively. In the case of the indirect formulation ($\epsilon = 1$), the FDC value is 0.8478, intermediate between $\epsilon = 3$, and $\epsilon = 4$. These results indicate that the indirect formulation can attain FDC values comparable to those of the direct formulation, but without suffering from some of the problems of the latter. Actually, the high FDC values for the direct formulation are compensated by two related facts, namely that there is a drift force towards low-fitness regions as mentioned in Sect. 4.1, and that the number of local optima is higher for low values of the local radius, specially in the high-fitness region.

4.3. Result Analysis

Our analysis indicates that the high irregularity of the neighborhood structure for the direct formulation introduces a drift force towards low-fitness regions of the search space. This contrasts with other problems in which the drift force is beneficial, since it guides the search to high-fitness regions (see [8]). The indirect formulation that we have considered does not have this drawback, and hence would be in principle more amenable for conducting local search in it. The fact that fitness-distance correlation is very similar in both cases also support this hypothesis. The computational cost is another issue to be considered. Navigating through the space of feasible solutions in the direct representation is very expensive from a computational point of view [10]. An indirect approach, or a direct approach in which the requirement of feasibility were dropped, is more adequate for search methods based on the iterative generation of solutions. These two approaches will be precisely considered in the following section.

5. Hybrid Approaches Based on Scatter Search

In this section we consider two hybrid algorithms that incorporate ideas from the method described in Sect. 3 (i.e., the EA based on greedy randomized adaptive search procedures (GRASP)), and tabu-based local search methods (TS), constraint programming and scatter search (SS). This last technique, described in the following, is a generic template which is thoroughfully described in [32].

5.1. A Basic View of Scatter Search

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 [32]. 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. These are also distinctive features of memetic algorithms (MAs) [35, 36, 38, 41]. Actually, in a broad sense, SS might be considered a particular case of MA. As in nature, the convergent evolution of the above-mentioned algorithmic features thus emphasizes their practical usefulness.

Fig.4 shows the diagram of a generic SS algorithm. As it can be seen, the algorithm is based on the iterative application of a collection of search procedures on a pool of solutions (the reference set), much like it is done in EAs (actually, SS can be safely characterized as an EA). More precisely, the following components can be identified in the algorithm:

- A *diversification generation method* for generating a collection of raw solutions, possibly using some initial solution as "seed".
- An *improvement method* for enhancing the quality of raw solutions.
- A *reference set update method* for building the reference set from the initial set of solutions generated, and for maintaining it by incorporating some solutions produced in subsequent steps.

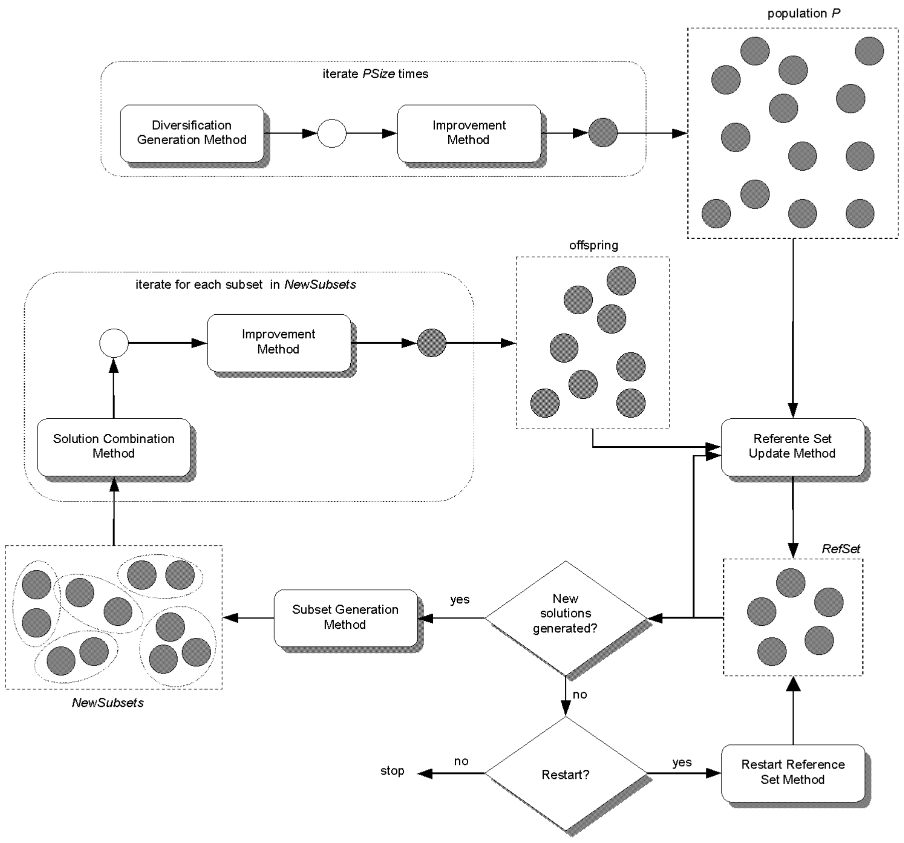


Figure 4: Sketch of the scatter search algorithm. Solutions for the problem considered are represented as circles. White circles are used to represent "raw" solutions, as obtained from the application of the diversification method or the solution combination method; as to dark circles, they represent "improved" solutions obtained by applying the improvement method to the former solutions.

- A *subset generation method* for selecting solutions from the reference set, and arranging them in small groups (pairs, triplets, or larger groups) for undergoing combination.
- A *solution combination method* for creating new raw solutions by combining the information contained in a certain group of solutions.
- A *restart reference set method* for refreshing the reference set once it has been found to be stagnated. This can be typically done by using the diversification generation method plus the improvement method mentioned above, but other strategies might be considered as well.

The specification of a particular SS algorithm is completed once the items above are detailed. The next subsections will be devoted to this purpose.

5.2. Scatter Search for the Golomb Ruler Problem

Our SS 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 HEAGRASP. The direct approach is considered in the stages of recombination and local improvement; particularly, the local improvement method is based on a tabu search (TS) algorithm described in Sect. 5.2.2. In the following we describe the instantiation of each component of the algorithm.

5.2.1. 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 mentioned in Sect. 3.2. To be precise, solutions are constructed by generating random parameter vectors $\langle r_1, \cdots, r_{n-2}\rangle$, $r_i \in [1..n]$, and feeding them to the GRASP-decoder. 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 $a_n < L$, 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. 5.2.5), which guarantees compliance with the mentioned constraint.

5.2.2. 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 $a_n < L$. 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 $\upsilon_\sigma(d)$ of a distance d in a n marks ruler σ is the number of times distance d appears between two marks in the ruler σ beyond its allowed occurrences, i.e.,

$$\upsilon_\sigma(d) = \max(0, \#\{d_{ij} = d \mid 1 \leqslant i < j \leqslant n\} - 1) \tag{4}$$

where $d_{ij} = a_j - a_i$. The overall violation $\upsilon(\sigma)$ of a n marks ruler σ is simply the sum of the violations of its distances d, i.e., $\upsilon(\sigma) = \sum_{d \in D} \upsilon_\sigma(d)$, where $D = \{d_{ij} \mid 1 \leqslant i < j \leqslant n\}$.

The moves in the local search consist of changing the value of a single mark. Since marks are ordered, a mark a_x can only take a value in the interval $I_\sigma(x) = [a_{x-1}+1, a_{x+1}-1]$. 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 a_1 is fixed to 0, and a_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

$$\mathcal{M}^+(\sigma,k) = \{(x,p) \in \mathcal{M}(\sigma) \mid \neg tabu(x,p,k)\}. \tag{5}$$

where $tabu(x,p,k)$ holds if the assignment $a_x \leftarrow p$ 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

$$\mathcal{M}^*(\sigma,\sigma^*) = \{(x,p) \in \mathcal{M}(\sigma) \mid \upsilon(\sigma[a_x \leftarrow p]) < \upsilon(\sigma^*)\} \tag{6}$$

where $\sigma[a_x \leftarrow p]$ denotes the ruler σ where variable a_x is assigned to p. To intensify the search, the current solution is reinitialized to the initial ruler σ_0 (in the current TS run) whenever no improvement in the number of violations took place for *maxStable* iterations. The algorithm returns the best solution σ^* found. Fig. 5 shows the complete pseudocode of the TS algorithm.

5.2.3. Reference Set Update Method. 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

```
1.  TS($\sigma_0$)
2.      $tabu \leftarrow \{\}$;
4.      $\sigma^* \leftarrow \sigma_0$;
5.      $k \leftarrow 0$;
6.      $s \leftarrow 0$;
7.      while $k \leqslant maxIter$ & $v(\sigma) > 0$ do
8.          select $(x,p) \in \mathcal{M}^+(\sigma,k) \cup \mathcal{M}^*(\sigma,\sigma^*)$ minimizing $v(\sigma[a_x \leftarrow p])$;
9.          $\tau \leftarrow$ RANDOM([4,100]);
10.         $tabu \leftarrow tabu \cup \{\langle x, p, k+\tau \rangle\}$;
11.         $\sigma \leftarrow \sigma[a_x \leftarrow p]$;
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 $\sigma^*$;
```

Figure 5: Pseudocode of the TS algorithm

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 invoked[1]. 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.

5.2.4. Subset Generation Method. 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 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.

5.2.5. Solution Combination Method. The combination of solutions is performed using a procedure that bears some resemblance with the GRASP-decoding mentioned in Sect. 5.2.1. 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 from either of the parents; finally, the method ensures that the $a_n < L$ 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 $x < L$ [2]. Then, starting from $a_1 = 0$, a new mark x is chosen at each step i such that (i) $a_{i-1} < x$, (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 $\sum_{j=1}^{i-1} \upsilon_\sigma(x - a_j)^2 + (x - a_{i-1})$, 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.

5.3. Experimental Results

To evaluate our hybrid (memetic) approach, a set of experiments for problem sizes ranging from 10 marks up to 16 marks has been realized (instances below 10 marks are too easy, so they have not been considered). 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 [25], 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_{LS} for applying LS (i.e., TS) to each solution. The values $p_{LS} \in \{0.1, 0.2, 0.4, 0.6, 0.8, 1.0\}$ have been considered. All algorithms were run 20 times until an optimal solution was 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_{LS}. This is a fair measure since the computational cost is dominated by the number of TS invocations.

Table 4 reports the experimental results for the different instances considered. Row SSxx corresponds to the execution of our algorithm with a local improvement rate of $p_{LS} = xx$. The table reports the relative distance (percentage) to the known

Table 4: Relative distances to optimum for different probabilities of the SS algorithm and the algorithms GROHEA and HEAGRASP. Globally best results (resp. globally best median results) for each instance size are shown in boldface (resp. underlined).

		number of marks						
		10	11	12	13	14	15	16
HEAGRASP	Best	**0**	2.8	10.6	4.7	6.3	7.3	6.8
	Median	$\underline{0}$	2.8	11.8	7.5	9.4	9.9	11.3
GROHEA	Best	**0**	**0**	**0**	**0**	3.1	4.6	5.6
	Median	$\underline{0}$	$\underline{0}$	7.1	5.6	7.1	8.6	10.2
SS1.0	Best	**0**	**0**	**0**	**0**	1.6	**0**	4.0
	Median	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	2.4	4.0	6.2
SS0.8	Best	**0**	**0**	**0**	**0**	0.8	1.3	2.3
	Median	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{1.6}$	$\underline{3.3}$	$\underline{5.6}$
SS0.6	Best	**0**	**0**	**0**	**0**	0.8	**0**	2.8
	Median	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{1.6}$	4.0	6.2
SS0.4	Best	**0**	**0**	**0**	**0**	**0**	1.3	**1.1**
	Median	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{1.6}$	4.0	$\underline{5.6}$
SS0.2	Best	**0**	**0**	**0**	**0**	**0**	0.7	3.4
	Median	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{1.6}$	4.0	6.2
SS0.1	Best	**0**	**0**	**0**	**0**	**0**	0.7	3.4
	Median	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{1.6}$	$\underline{3.3}$	$\underline{5.6}$

optimum for the best and median solutions obtained. The table also shows the results obtained by HEAGRASP. As to algorithm GROHEA [13], 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, 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.

The results are particularly impressive. Firstly, observe that our hybrid 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 SS algorithm. In fact, the median values obtained by the SS algorithm for these instances correspond exactly to their optimal solutions. Comparatively, the results are even better in larger OGR instances: our SS 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 SSxx 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.

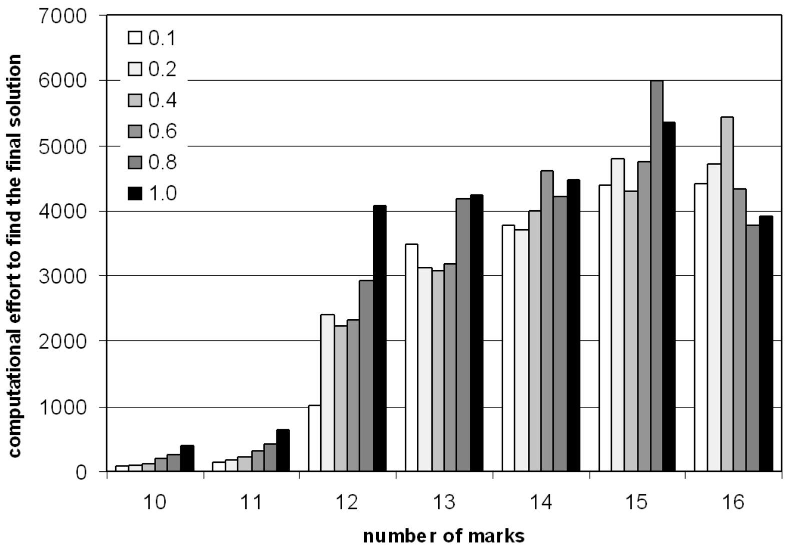


Figure 6: Computational effort (measured in number of TS invocations) to find the best solution.

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 again. Except in three head-to-head comparisons for 14 marks ($p_{LS} = 1.0$ vs $p_{LS} = 0.8$ and $p_{LS} = 0.1$, and $p_{LS} = 0.4$ vs $p_{LS} = 0.1$), there is no statistically significant difference (at the standard 0.05 level) in any instance size for the different values of p_{LS}. 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. 6. 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). A statistical comparison between the computational cost of SSxx for a given instance size indicates that the differences are almost always significant for the lower range of sizes (up to 12 marks), and become non-significant as the size increases. For 16 marks, there is just one case of statistically significant difference of computational cost ($p_{LS} = 0.4$ vs $p_{LS} = 0.8$). Since the small values of p_{LS} 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_{LS} in the larger instances, it seems that values $p_{LS} \in \{0.1, 0.2\}$ are advisable.

5.4. Scatter Search + CP + Clustering

The algorithm presented in the previous sections yields very impressive results, however, we want to pursue it further. There are two aspects (among others) that we can improve very straightforwardly. First, we realized that Constraint Programming can be of help at some point. Second, we believe that diversity in the population is almost as important as the quality of it. Basically, our last proposal is an improvement over the hybrid algorithm presented in the previous section. It relies on clustering to achieve diversity in the reference set and complete search to attempt to find optimal rulers at the recombination step. These enhancements produce superior results as it will be shown later (e.g., the new hybrid is now capable of solving rulers up to 16 marks). Let us then introduce the new features incorporated into our algorithm:

5.4.1. Solution Combination by Complete Search. Recombination methods are usually introduced in order to generate new high quality and diverse individuals. Our current recombination mechanism achieves these goals, however what we pursue here is something different. We are trying to generate optimal solutions with this operator.

We have been dealing with values of marks through all this research. Now we turn to look into the distances between marks. We realized that a complete search procedure that incorporates propagation techniques would be perfectly suited to search for a solution when fed with the appropriate distances. Complete search procedures tend to be very inefficient with very large search spaces, however we can limit that in this case by only taking into account the distances between marks of the two individuals to be combined. For example, imagine we have the two rulers of 9 marks to be combined:

$\sigma_1 = \langle$ 0 1 5 13 23 34 47 50 55 $\rangle$
$\sigma_2 = \langle$ 0 2 7 11 12 24 30 40 47 $\rangle$

The distances between marks of both rulers are

[1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13]

Note that we only consider distances between two consecutive marks i, j where $i + 1 = j$. To fully characterize the problem we need to take all the distances into account, however these are the distances we are going to restrict the search to. More precisely, we use those distances to feed a complete search procedure whose goal is to quickly attempt to generate valid (hopefully optimal) rulers. In order to do this we need to formulate the problem as a CSP. Now, the variables are distances D_{ij} between marks (where $i < j$). The domain of the variables D_{ij} where $i + 1 = j$ is reduced to the values previously shown. The rest of the variables can take any value in $[1..m]$ where m is the length of the ruler (note that the value of m is not important as we will soon clarify). The set of constraints is as follows:

$$\forall i, j, k, l\ (i \neq k, j \neq l)\ :\ D_{ij} \neq D_{kl} \tag{7}$$

Since we are going to execute this procedure within the SS cycle, we can take advantage of dynamic information, such as the length L of the shortest valid ruler in the population. This is explicitly indicated in the procedure by introducing a constraint:

$$min(l_k) < L \tag{8}$$

where $min(l_k)$ is the minimum length possible for a partial ruler k.

5.4.2. Empirical Observations. We have now fully characterized the problem our complete search procedure is going to be dealing with. However we found two options at this point: (1) use a CP solver, or (2) take advantage of the data structures already implemented in our algorithm. The first option implies we can plug a black-box to our SS algorithm to which we pass a set of distances and then expect a solution or a confirmation that no valid solution can be found. We can take advantage of efficient propagation techniques and sophisticated heuristics.

On the other hand, taking advantage of the structures already implemented, we can focus on instantiating only variables D_{ij} corresponding to distances between consecutive marks; if we instantiate the variables in the same order as they would physically appear in the ruler, we can easily calculate the rest of the distances and thus, check the validity of the partial solution very quickly. This can be viewed as limiting the search variables to the ones representing distances between consecutive marks and using a lexicographic variable ordering heuristic. Note that in this case we do not need to worry about the value of m, the upper bound in the value of the non-consecutive distances, since we are not focusing the search on them, but only on the consecutive ones.

Both approaches were tested and we found that the latter was consistently faster than the former; as it allows us to ignore non-consecutive distance variables. Still, the reduction of the search space was not enough to yield a very fast mechanism. Remember that we implement this procedure as a combination operator, and thus, we cannot devote more than a few seconds to it.

In our experimental tests we discovered that, when reducing the search space to consider only the consecutive distances of the optimal ruler, the procedure was able to find a solution in less than a second for up to 14 marks, and less than 5 seconds for 15 and 16 marks. However, introducing more distances slowed down the process exponentially, and if within those distances a valid solution was not possibly found, the time cost grew immensely. After a few more experiments we designed the next mechanism:

1. Select the two rulers to be combined.
2. Calculate the consecutive distances having into account that some distances might be repeated; for example, in two 15 mark rulers (14 consecutive distances) we typically find around 20 different distances.

3. Randomly select $n + 3$ distances from the previous step, being n the number of marks in the ruler (i.e., 4 distances more than needed).

4. Run the complete search procedure with those distances, with a time limit of 4 seconds.

5. If a valid solution was found, return it, if not call the old recombination mechanism.

Note that this procedure can also find near optimal rulers if the chosen distances allow it. Obviously, we can be missing some potential optimal rulers by randomly selecting $n+3$ distances, but we found it to be the best trade-off between time and efficiency. That is, if there was a valid solution for the given distances, the complete search procedure would almost always find it within the given time limit.

5.4.3. Diversity in the Population: Clustering. We also realized that the population got stuck very frequently. The solutions provided by the LS mechanism were of high quality, and thus converged very quickly to the same region of the search space. Restarts were required to drive the search towards different regions. Since the population was selected in an elitist fashion, the algorithm was often unable to generate better individuals that could be included in the reference set.

Diversity is thus a key aspect of a population. It is very important to be able to generate individuals in different regions of the search space while maintaining a relatively high quality in all of them. In this sense we directed our efforts towards implementing a clustering algorithm. Clustering deals with finding a structure in a collection of unlabelled data, and it can be considered the most important unsupervised learning mechanism; a loose definition of clustering could be *"the process of organizing objects into groups whose members are similar in some way"*. A cluster is therefore a collection of objects which are "similar" among them and are "dissimilar" to the objects belonging to other clusters.

In our population we have individuals that are vectors of marks, however, we are going to transform them into distances between marks as in the previous subsection. Our goal is thus to group individuals with similar sets of distances in the same cluster. The algorithm for clustering is very simple; imagine we consider τ clusters:

1. Transform the vectors of marks into vectors of distances (actually, in binary vectors that indicate whether a distance is included in the individual or not). For example, the individual representing a 9 marks ruler

   $\sigma_1 = \langle\ 0\ 1\ 5\ 13\ 23\ 34\ 47\ 50\ 55\ \rangle$

   will be transformed into

   $\mathrm{x}(\sigma_1) = [\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ \ldots\ ]$

   where a 1 in $\mathrm{x}(\sigma_1)[i]$ indicates that the distance i is included in the individual σ_1, where i is in the range from to three times the number of marks; this limit

l was imposed after the observation that optimal rulers try to incorporate the lowest different distances between consecutive marks so that the ruler's length is minimized.

2. Calculate τ random centroids. The centroids are the vectors that represent the clusters, their central points. Thus, the centroids are vectors of the same length as the individuals that characterize a cluster. If we already had the vectors separated in clusters, the centroid of a cluster k_i would be a vector:

$$centroid(k_i) = [\frac{a_1}{\mid k_i \mid}\frac{a_2}{\mid k_i \mid}\cdots\frac{a_l}{\mid k_i \mid}] \tag{9}$$

   where $\mid k_i \mid$ is the number of vectors in cluster k_i and a_j is the number of vectors in cluster k_i in which thet j-th bit is set to 1. Since we have no clusters yet we calculate the centroids randomly.

3. Assign every vector to its nearest centroid, creating thus the clusters. The distance measure we use is that of the cosine of the angles formed by the vector and the centroid.

4. Recalculate the centroids with the now real information of the vectors in the clusters.

5. Repeat steps 3 and 4 until no centroid is changed in step 4 or until a maximum number of iterations is reached, in our case 10.

This procedure thus follows the conspicuous k-means algorithm [33]. Upon application of this algorithm, the population is divided into τ clusters. This fact itself does not ensure diversity in the reference set. To maintain a high degree of diversity without harnessing its quality we rank the vectors in every cluster and then select the best ω individuals from each cluster, and we include them in the Reference Set.

Note that this process is relatively time consuming, and thus, it is only performed for the initial population and after a restart. However, at any generation, the algorithm updates the reference set in a way that the premise *the best ω individuals of each cluster are maintained in the reference set* is satisfied.

5.5. Experimental Results on the CP-Based Hybrid Proposal

In this section we show results for our hybrid algorithm after the incorporation of the new features. The experiments have been performed on rulers from 14 up to 16 marks, and with probability $p_{LS} = 0.1$, which we found to be one of the most consistent ones after the previous experiments. Regarding the diversity mechanism, we have performed three different sets of experiments varying the values of the clustering parameters. These different sets of parameters are: $\tau = 5$, $\omega = 4$; $\tau = 10$, $\omega = 2$ and $\tau = 20$, $\omega = 1$. The remaining parameters are the same as in Sect. 5.3.

Table 5 depicts the results for these new experiments and a comparison of the results presented in Sect. 5.3 for $p_{LS} = 0.1$. As a first result we can see that for

Table 5: Relative distances to the optimum for SS0.1 and the improved algorithm SS+τ-ω. Globally best results (resp. globally best median results) for each instance size are shown in boldface (resp. underlined).

	14		15		16	
	best	median	best	median	best	median
SS0.1	**0**	1.6	0.7	$\underline{3.3}$	3.4	$\underline{5.6}$
SS+5-4	**0**	$\underline{0}$	**0**	$\underline{3.3}$	**0**	$\underline{5.6}$
SS+10-2	**0**	1.6	1.3	4.6	5.1	7.3
SS+20-1	**0**	0.8	0.7	5.2	3.4	7.3

14 marks the algorithm SS+5-4 always finds the optimal solution, which did not happen with SSxx. Secondly (and maybe more important), we are now able to solve the 16 marks ruler. Also note that SS+5-4 seems to dominate the rest of the instances of SS+τ-ω, perhaps because it corresponds to the optimal balance between quality and diversity.

6. Summary, Conclusions and Future Work

Finding Golomb rulers is an extremely challenging optimization problem with many practical applications that has been approached by a variety of search methods in recent years. It combines hard and dense feasibility constraints and an optimization function to minimize the length of the ruler. In this paper we have proposed a battery of local-search based algorithms that find optimal or near-optimal Golomb rulers at a reasonable computational cost.

Initially, we have described 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 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. This algorithm, deeply described in [9], has also been the starting point of the subsequent evolutionary hybrid algorithms.

Then, we have tried to divulge some light on the question of what makes a problem hard for a certain search algorithm for the Golomb Ruler Problem. This has been done via an analysis of the fitness landscape of the problem, as it was done by the authors in [10]. Our analysis indicates that the high irregularity of the neighborhood structure for the direct formulation introduces a drift force towards low-fitness regions of the search space. The indirect formulation that we have considered does not have this drawback, and hence would be in principle more amenable for conducting local search in it.

We have also proposed a memetic (hybrid) approach (MA) 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 Lamarckianism achieve the best tradeoff between computational cost and solution quality.

Later, we have introduced several improvements to the previous algorithm to yield outstanding results: we are able to solve a 16 marks ruler and to consistently solve every 14 marks rulers in our testbed. This last algorithm is based on a Scatter Search template and includes a complete search inherited technique to combine individuals, and a clustering procedure which we apply to our population in order to achieve a higher degree of diversity.

The algorithm tested using different sets of parameters referred to the clustering mechanism is consistently superior to the previous algorithm without the improvements.

Related to the paragraph above it arises the issue of automatic parameter adjustment. Our algorithm has a few parameters that have been manually tuned throught out the research. However, there are several interesting techniques that have appeared in the literature in the last years. In the future, it would be very clever to try to include them in our algorithm and study their performance. Examples of these are [34] and [1].

We are currently exploring alternatives for some of the operators used in this last 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. Defining appropriate distance measures in this context (and indeed, checking their usefulness in practice) will be the subsequent step.

As for the final hybrid resulting of the introduction of the new improvements, there is a very obvious observation: the SS and the LS deal with marks, while the Clustering and Complete Search deal with distances. We plan to make it uniform and possibly implement all the techniques so they can deal with distances. More efficient clustering techniques are also worth being studied.

Acknowledgements.

The work of first and third authors was partially supported by Spanish MCyT under contracts TIN2004-7943-C04-01 and TIN2005-08818-C04-01.

Notes

1. 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.
2. It might happen that the number of such marks were not enough to build a new ruler. In that case, a dummy solution with length ∞ (that is, the worst possible value) is returned.

References

1. B. Adenso-Diaz and M. Laguna. Fine tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114, 2006.
2. W.C. Babcock. Intermodulation interference in radio systems. *Bell Systems Technical Journal*, pages 63–73, 1953.
3. R. Barták. Practical constraints: A tutorial on modelling with constraints. In J. Figwer, editor, *5th Workshop on Constraint Programming for Decision and Control*, pages 7–17, Gliwice, Poland, 2003.
4. F. Biraud, E. Blum, and J. Ribes. On optimum synthetic linear arrays with applications to radioastronomy. *IEEE Transactions on Antennas and Propagation*, 22(1):108–109, 1974.
5. J. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6:154–160, 1994.
6. G.S. Bloom and S.W. Golomb. Aplications of numbered undirected graphs. *Proceedings of the IEEE*, 65(4):562–570, 1977.
7. K.D. Boese, A.B. Kahng, and Muddu S. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16:101–113, 1994.
8. C. Bierwirth, D.C. Mattfeld, and J.-P. Watson. Landscape regularity and random walks for the job shop scheduling problem. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3004 of *Lecture Notes in Computer Science*, pages 21–30, Berlin, 2004. Springer.
9. C. Cotta and A.J. Fernández. A hybrid GRASP - evolutionary algorithm approach to Golomb ruler search. In Xin Yao et al., editors, *Parallel Problem Solving From Nature VIII*, number 3242 in Lecture Notes in Computer Science, pages 481–490, Berlin Heidelberg, 2004. Springer.
10. C. Cotta and A.J. Fernández. Analyzing fitness landscapes for the optimal golomb ruler problem. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3248 of *Lecture Notes in Computer Science*, pages 68–79, Berlin Heidelberg, 2005. Springer.
11. D. Diaz and P. Codognet. GNU Prolog: beyond compiling Prolog to C. In E. Pontelli and V. Santos Costa, editors, *2nd International Workshop on Practical Aspects of Declarative Languages (PADL'2000)*, volume 1753 of *Lecture Notes in Computer Science*, pages 81–92, Boston, USA, 2000. Springer.
12. A.K. Dewdney. Computer recreations. *Scientific American*, pages 14–21, 1986.
13. Iván Dotú and Pascal Van Hentenryck. A simple hybrid evolutionary algorithm for finding golomb rulers. In D. Corne et. al., editor, *Congress on Evolutionary Computation Conference (CEC2005)*, volume 3, pages 2018–2023, Edinburgh, Scotland, 2005. IEEE.
14. A. Dollas, W. T. Rankin, and D. McCracken. A new algorithm for Golomb ruler derivation and proof of the 19 mark ruler. *IEEE Transactions on Information Theory*, 44:379–382, 1998.
15. B. Feeney. Determining optimum and near-optimum Golomb rulers using genetic algorithms. Master thesis, Computer Science, University College Cork, October 2003.
16. T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
17. R.J.F. Fang and W.A. Sandrin. Carrier frequency assignment for non-linear repeaters. *Comsat Technical Review*, 7:227–245, 1977.

18. C. Giraud-Carrier. Unifying learning with evolution through baldwinian evolution and lamarckism: A case study. In H-J. Zimmermann, G. Tselentis, M. van Someren, and G. Dounias, editors, *Advances in Computational Intelligence and Learning: Methods and Applications*, pages 159–168. Kluwer Academic Publishers, 2002.
19. P. Galinier, B. Jaumard, R. Morales, and G. Pesant. A constraint-based approach to the Golomb ruler problem. In *3rd International Workshop on Integration of AI and OR Techniques (CP-AI-OR'2001)*, 2001.
20. D.E. Goldberg and R. Lingle Jr. Alleles, loci and the traveling salesman problem. In J.J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms*, Hillsdale, 1985. Lawrence Erlbaum Associates.
21. F. Glover. Tabu search – part i. *ORSA Journal of Computing*, 1(3):190–206, 1989.
22. F. Glover. Tabu search – part ii. *ORSA Journal of Computing*, 2(1):4–31, 1989.
23. F. Glover. A template for scatter search and path relinking. *Lecture Notes in Computer Science*, 1363:13–54, 1997.
24. M. Garry, D. Vanderschel, et al. In search of the optimal 20, 21 & 22 mark Golomb rulers. GVANT project, `http://members.aol.com/golomb20/index.html`, 1999.
25. C. Houck, J.A. Joines, M.G. Kay, and J.R. Wilson. Empirical investigation of the benefits of partial lamarckianism. *Evolutionary Computation*, 5(1):31–60, 1997.
26. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L.J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, 1995. Morgan Kaufmann Publishers.
27. A.K. Jain, N.M. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.
28. T. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search.* Phd thesis, Santa Fe Institute, University of New Mexico, Alburquerque, May 1995.
29. B.A. Julstrom. Comparing darwinian, baldwinian, and lamarckian search in a genetic algorithm for the 4-cycle problem. In S. Brave and A.S. Wu., editors, *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pages 134–138, Orlando, FL, 1999.
30. T. Klove. Bounds and construction for difference triangle sets. *IEEE Transactions on Information Theory*, 35:879–886, July 1989.
31. E.L. Lehmann and H.J.M. D'Abrera. *Nonparametrics: Statistical Methods Based on Ranks.* Prentice-Hall, Englewood Cliffs, NJ, 1998.
32. M. Laguna and R. Martí. *Scatter Search. Methodology and Implementations in C.* Kluwer Academic Publishers, Boston MA, 2003.
33. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.
34. L. Paquete M. Birattari, T Stutzle and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W.B. Lagndon et al., editor, *Genetic and Evolutionary Computation Conference (GECCO)*, pages 11–18, San Francisco, CA, 2002. Morgan Kaufmann.
35. P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers, Boston MA, 2003.
36. P. Moscato and C. Cotta. Memetic algorithms. In T. González, editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 22. Taylor & Francis, 2006.
37. D. McCracken. Minimum redundancy linear arrays. Senior thesis, Duke University, Durham, NC, January 1991.
38. P. Moscato, C. Cotta, and A. Mendes. Memetic algorithms. In G.C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, pages 53–85. Springer, Berlin Heidelberg, 2004.
39. B. Manderick, M. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150, San Mateo CA, 1991. Morgan Kaufmann.

40. P. Mirchandani and R. Francis. *Discrete Location Theory.* Wiley-Interscience, 1990.
41. P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, Maidenhead, Berkshire, England, UK, 1999.
42. OGR project. `http://www.distributed.net/ogr/`, on-going since September 14, 1998.
43. M. Prais and C.C. Ribeiro. Parameter variation in GRASP procedures. *Investigación Operativa*, 9:1–20, 2000.
44. M. Prais and C.C. Ribeiro. Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176, 2000.
45. S. Prestwich. Trading completeness for scalability: Hybrid search for cliques and rulers. In *CPAIOR-01*, pages 159–174, Ashford, Kent, England, 2001.
46. F.B. Pereira, J. Tavares, and E. Costa. Golomb rulers: The advantage of evolution. In F. Moura-Pires and S. Abreu, editors, *Progress in Artificial Intelligence, 11th Portuguese Conference on Artificial Intelligence*, number 2902 in Lecture Notes in Computer Science, pages 29–42, Berlin Heidelberg, 2003. Springer.
47. N.J. Radcliffe. Equivalence class analysis of genetic algorithms. *Complex Systems*, 5:183–205, 1991.
48. W.T. Rankin. Optimal Golomb rulers: An exhaustive parallel search implementation. Master thesis, Duke University Electrical Engineering Dept., Durham, NC, December 1993.
49. J.P. Robinson and A.J. Bernstein. A class of binary recurrent codes with limited error propagation. *IEEE Transactions on Information Theory*, 13:106–113, 1967.
50. C. Reeves. Landscapes, operators and heuristic search. *Annals of Operational Research*, 86:473–490, 1999.
51. J. Robbins, R. Gagliardi, and H. Taylor. Acquisition sequences in PPM communications. *IEEE Transactions on Information Theory*, 33:738–744, 1987.
52. M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, Boston MA, 2003.
53. W. Schneider. Golomb rulers. MATHEWS: The Archive of Recreational Mathematics, `http://www.wschnei.de/number-theory/golomb-rulers.html`, 2002.
54. J.B. Shearer. Some new optimum Golomb rulers. *IEEE Transactions on Information Theory*, 36:183–184, January 1990.
55. J. B. Shearer. Golomb ruler table. Mathematics Department, IBM Research, `http://www.research.ibm.com/people/s/shearer/grtab.html`, 2001.
56. S.W. Soliday, A. Homaifar, and G.L. Lebby. Genetic algorithm approach to the search for Golomb rulers. In L.J. Eshelman, editor, *6th International Conference on Genetic Algorithms (ICGA'95)*, pages 528–535, Pittsburgh, PA, USA, 1995. Morgan Kaufmann.
57. B.M. Smith and T. Walsh. Modelling the Golomb ruler problem. In *Workshop on non-binary constraints (IJCAI'99)*, Stockholm, 1999.
58. S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In D.F. Jones, editor, *6th Intenational Congress on Genetics*, volume 1, pages 356–366, 1932.