

A simple Hybrid Evolutionary Algorithm for Finding Golomb Rulers

Iván Dotú

Departamento De Ingeniería Informática
Universidad Autónoma de Madrid

Pascal Van Hentenryck

Brown University, Box 1910
Providence, RI 02912

Abstract- 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. This paper presents a hybrid evolutionary algorithm to find near-optimal Golomb rulers in reasonable time. The algorithm, which is conceptual simple and uses a natural modeling, focuses on feasibility, finding near-optimal rulers indirectly. It significantly outperforms earlier (hybrid) evolutionary algorithms and compares favorably with hybridizations of local search and constraint programming. In particular, the algorithm quickly finds optimal rulers with up to 11 marks and isolates optimal rulers with up to 14 marks in reasonable time. It also finds near-optimal rulers for up to 16 marks quickly.

1 Introduction

Finding Golomb rulers is an extremely challenging combinatorial problem which has received considerable attention over the last decades. An 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

$$m_j - m_i \quad (1 \leq i < j \leq n) \quad (1)$$

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 . An n -mark Golomb ruler is *optimal* if there exists no n -mark Golomb ruler of smaller length.

Golomb rulers have applications in a wide variety of fields including radio communications ([Blum, Biraud, & Ribes 1974, Hayes 1998]), x-ray crystallography ([Bloom, & Golomb 1977]), coding theory ([Dolas, Rankin, & McCracken 1998, Klove 1989]), and radio astronomy. Moreover, because of its highly combinatorial nature,¹ it has become a standard benchmark to evaluate and compare a variety of search techniques. In particular, genetic algorithms, constraint programming, local search, and their hybridizations have all been applied to the problem of finding Golomb rulers (e.g., [Cotta & Fernández 2004, Feeney 2003, Pereira, Tavares, & Costa 2003, Prestwich 2001, Smith, Stergiou, & Walsh 1999, Soliday, Homaifar, & Lebby 1995]).

This paper proposes a novel hybrid evolutionary algorithm for finding near-optimal Golomb rulers in reasonable time. The algorithm embeds a local search into a genetic algorithm and outperforms earlier genetic algorithms, as well as constraint programming algorithms and their hybridizations with local search. In particular, the algorithm quickly finds optimal rulers for up to 13 marks and was able to find optimal rulers for 14 and 15 marks, which is clearly out of scope for the above mentioned algorithms. The algorithm also finds near-optimal rulers in reasonable time, clearly indicating the effectiveness of hybrid evolutionary algorithms on this highly combinatorial application. Of particular interest is the conceptual simplicity and elegance of the algorithm.

Even though there are solutions for higher number of marks for other complete search approaches, evolutionary algorithms have the advantage of providing good quality solutions in a short period of time. This is a main contribution of this research as well, providing high quality solutions (improving all previous evolutionary approaches) in a few seconds or minutes.

The main technical contribution of the novel hybrid evolutionary algorithm is its focus on feasibility. Indeed, the main step of the evolutionary algorithm is to find a Golomb ruler of a specified length (or smaller), using constraint violations to guide the search. Near-optimal rulers are obtained indirectly by solving a sequence of feasibility problems.

The rest of this paper starts by a brief overview of related work. It then presents the local search and the hybrid evolutionary algorithm for finding Golomb rulers of a specified length, before generalizing the algorithm to find near-optimal rulers and concluding.

2 Related Work

There is a wide spectrum of approaches for finding Golomb rulers and it is outside the scope of this paper to review them all. This section focuses on evolutionary methods and readers are recommended to consult [Dolas, Rankin, & McCracken 1998] for an extensive review of constructive methods and [Smith, Stergiou, & Walsh 1999, Prestwich 2001] for constraint programming approaches and their hybridizations with local search. The approach in [Prestwich 2001] is particularly interesting and will also be used for comparison purposes.

[Soliday, Homaifar, & Lebby 1995] proposed a genetic algorithm whose decision variables are the distances between each mark (segments). The initial population was designed to include length 1 in the ruler and genetic op-

¹The search for a 19-mark Golomb ruler took approximately 36,200 CPU hours on a Sun Sparc workstation using a very specialized algorithm [Dolas, Rankin, & McCracken 1998].

erators maintain this property. Mutations consist of permutating the segment order and changing the segment lengths. Crossovers split rulers into several parts which are then recombined. The algorithm uses two fitness criteria: the overall length of the ruler; and the number of repeated segments. This approach was extended and refined by [Pereira, Tavares, & Costa 2003] who used random keys [Bean 1994] to encode the segments. They also introduced a heuristic which yields some improvement in the quality of the solutions.

[Feeney 2003] studied various hybridizations of genetic algorithms and local search. More precisely, he studied three different genetic algorithms: a pure genetic algorithm, an hybridization with local search and Baldwinian learning, and an hybridization with local search and Lamarkian learning. Although interesting, these methods achieved poor results in terms of performance.

More recently, [Cotta & Fernández 2004] proposed an hybridization of genetic algorithms with a reactive GRASP (Greedy Randomized Adaptive Search Procedure). This last algorithm outperforms earlier approaches and will be used to evaluate our algorithm.

3 Golomb Rulers of Fixed Lengths

This section describes hybrid evolutionary algorithm for finding Golomb rulers of specified lengths. It starts with the problem modeling, then describes the local search and the hybrid evolutionary algorithms, and concludes with the experimental results.

3.1 Modeling

The problem modeling in the hybrid evolutionary algorithm is natural and associates a decision variable m_x with every mark x . Given a ruler σ , i.e., an assignment of values to the decision variables, the value $\sigma(m_x)$ denotes the position of mark x within the ruler. Since the length l of the ruler is known in this section, the values $\sigma(m_1)$ and $\sigma(m_n)$ are fixed to 0 and l respectively. There are three kinds of constraints in the Golomb ruler:

1. The marks have different positions in the ruler.
2. The marks are ordered, i.e., $\sigma(m_i) < \sigma(m_{i+1})$.
3. The distances $d_{ij} = m_j - m_i$ ($j > i$) are all different.

The first two types of constraints are implicit in the algorithms presented in this paper: They are satisfied by the initial assignments and are preserved by local moves and genetic operators. The goal of the algorithms is thus to satisfy the third set of constraints.

To guide the search, the algorithms use 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.,

$$v_\sigma(d) = \max(0, \#\{d_{ij} = d \mid 1 \leq i < j \leq n\} - 1) \quad (2)$$

where $d_{ij} = \sigma(m_j) - \sigma(m_i)$. The violations $v(\sigma)$ of a n -mark ruler σ is simply the sum of the violations of its distances d , i.e.,

$$v(\sigma) = \sum_{d \in D} v_\sigma(d) \quad (3)$$

where $D = \{d_{i,j} \mid 1 \leq i < j \leq n\}$. Obviously, a ruler σ with $v(\sigma) = 0$ is a solution to the Golomb ruler problem.

3.2 The Tabu Search

We now turn to the tabu search algorithm for finding Golomb rulers of specified lengths.

The Neighborhood The moves in the local search consists of changing the value of a single mark. Since the marks are ordered, a mark x can only take a value in the interval

$$I_\sigma(x) = [\sigma(m_{x-1}) + 1, \sigma(m_{x+1}) - 1]. \quad (4)$$

As a consequence, the sets of possible moves is

$$\mathcal{M}(\sigma) = \{(x, p) \mid 1 < x < n \ \& \ p \in I_\sigma(x)\}. \quad (5)$$

Observe that $\sigma(m_1)$ and $\sigma(m_n)$ are fixed to 0 and l .

The Tabu Component The tabu component of the local search prevents a mark from being reassigned the same value for a number of iterations. The tabu list thus consists of a triplet $\langle x, p, i \rangle$, where x is a mark and 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 iteration (x, p) stays in the list, is dynamic and is randomly generated in the interval $[4, 100]$. For a ruler σ and an iteration k , the set of legal moves is thus defined as

$$\mathcal{M}^+(\sigma, k) = \{(x, p) \in \mathcal{M}(\sigma) \mid \neg \text{tabu}(x, p, k)\}. \quad (6)$$

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 would reduce the smallest number of violations found so far. In other words, if σ^* is the ruler with the smallest number of violations found so far, the neighborhood also includes the moves

$$\mathcal{M}^*(\sigma, \sigma^*) = \{(x, p) \in \mathcal{M}(\sigma) \mid v(\sigma[m_x \leftarrow p]) < v(\sigma^*)\} \quad (7)$$

where $\sigma[m_x \leftarrow p]$ denotes the ruler σ where variable m_x is assigned to p .

The Tabu-Search Algorithm We are now ready to present the basic local search algorithm GRLS. The algorithm, depicted in Figure 1, a tabu search with an intensification component. Lines 2-6 perform the initializations. In particular, the tabu list is initialized in line 2, the initial ruler is generated randomly in line 3, while lines 5 and 6 initialize the iteration counter k , and the stability counter s . The initial configuration σ randomly assigns values for all marks, satisfying the constraints that each mark is assigned to a different value and are ordered. Moreover, the position of the

```

1. GRLS( $n, l$ )
2.  $tabu \leftarrow \{\}$ ;
3.  $\sigma \leftarrow \text{RANDOMCONFIGURATION}(n, l)$ ;
4.  $\sigma^* \leftarrow \sigma$ ;
5.  $k \leftarrow 0$ ;
6.  $s \leftarrow 0$ ;
7. while  $k \leq \text{maxIt}$  &  $v(\sigma) > 0$  do
8.   select  $(x, p) \in \mathcal{M}^+(\sigma, k) \cup \mathcal{M}^*(\sigma, \sigma^*)$ 
     minimizing  $v(\sigma[m_x \leftarrow p])$ ;
9.    $\tau \leftarrow \text{RANDOM}([4, 100])$ ;
10.   $tabu \leftarrow tabu \cup \{ \langle x, p, k + \tau \rangle \}$ ;
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  $s > \text{maxStable}$  then
16.     $\sigma \leftarrow \sigma^*$ ;
17.     $s \leftarrow 0$ ;
18.     $tabu \leftarrow \{\}$ ;
19.  else
20.     $s++$ ;
21.     $k++$ ;

```

Figure 1: Algorithm GRLS for Finding Golomb Rulers

first mark is 0 and the position of the last mark is the length l of the ruler. The best ruler found so far σ^* is initialized to σ . The core of the algorithm are lines 7-21 which perform local moves for a number of iterations or until a solution is found. The local move is selected in line 8. The key idea is to select the best assignment in the neighborhood

$$\mathcal{M}^+(\sigma, k) \cup \mathcal{M}^*(\sigma, \sigma^*), \quad (8)$$

i.e., the non-tabu moves and those which improve the best ruler. Observe that the expression $v(\sigma[m_x \leftarrow p])$ represents the number of violations obtained after assigning p to mark x . The tabu list is updated in line 10, and the new ruler is computed in line 11. Lines 12-14 update the best ruler, while lines 15-18 specify the intensification component. The intensification component simply reinitializes the search from the best available ruler whenever no improvement in the number of violations took place for maxStable iterations. Note that the stability counter s is incremented in line 20 and reset to zero in line 14 (when a new best ruler is found) and in line 17 (when the search is restarted).

3.3 The Hybrid Evolutionary Algorithm

We now turn to the hybrid evolutionary (HE) algorithm for finding Golomb rulers of specified lengths. The algorithm maintains a population of rulers and performs a number of iterations until a solution is found. Each iteration selects two rulers in the population and, with some probabilities, crosses and/or mutates them. The two rulers so obtained replace their parents in the population. Each of these steps is now reviewed in more detail.

Selection Each iteration selects two rulers in the population (the parents). Two strategies were studied for selecting

the parents: a random strategy which randomly selects two rulers from the population and a ‘‘roulette wheel’’ strategy that biases the search toward rulers with fewer violations.

Crossover The HE algorithm uses a one-point crossover for crossing two rulers σ_1 and σ_2 . It selects a random number k in $1..n$. The first child is obtained by selecting the first k marks from σ_1 and the remaining $n - k$ marks from σ_2 . The second child is obtained in a similar fashion by swapping the role of the parents. There is a minor difficulty to address when crossing two rules: the two rulers may include the same markers. Consider the two parents

$$\begin{aligned} \sigma_1 &= \langle 0 \ 1 \ 5 \ 12 \ 23 \ 34 \ 37 \ 41 \ 44 \rangle \\ \sigma_2 &= \langle 0 \ 3 \ 6 \ 10 \ 16 \ 23 \ 39 \ 42 \ 44 \rangle \end{aligned}$$

and $k = 5$. Without extra care, the first child would be

$$\langle 0 \ 1 \ 5 \ 12 \ 23 \ / \ 23 \ 39 \ 42 \ 44 \rangle$$

repeating position 23. Instead, the crossover selects the last $n - k$ elements in σ_2 which are not found in σ_1 , giving

$$\langle 0 \ 1 \ 5 \ 12 \ 23 \ / \ 16 \ 39 \ 42 \ 44 \rangle.$$

The ruler is then ordered to obtain the first child

$$\langle 0 \ 1 \ 5 \ 12 \ 16 \ 23 \ 39 \ 42 \ 44 \rangle.$$

The second child is obtained in a symmetric way.

Mutation Mutations in the HE algorithm are performed by the local search GRLS. The best solution obtained by GRLS is the result of the mutation, unless this solution is already in the population. In this last case, the mutation is simply the ruler when the local search terminates. This design choice is motivated by the desire to preserve diversity during the search.

Restarting Policy The algorithm is restarted from scratch when the diversity of the population is too low. The restarting policy is based on the empirical observation that the population is not diverse enough when too many rulers have few violations. As a consequence, the HE algorithm restarts when more than half of the population has fewer violations than a specified threshold Υ . This strategy is only applied when the parents are selected using the ‘‘roulette wheel’’ strategy which has a tendency to decrease the diversity of the population significantly over time. In the following, we use $\text{diversity}(\Sigma)$ to denote the median violation in Σ and $v(\Sigma)$ to denote the smallest violation in Σ .

The Hybrid Algorithm We are now ready to present the HE algorithm GRHEA which is depicted in Figure 2. Lines 2-4 perform the initializations. In particular, the population is randomly generated in lines 2-3 and the generation counter g is initialized in line 4.

The core of the algorithm is in lines 5-21. They generate new generations of rulers for a number of iterations or until a solution is found. The new generation is initialized in line 7, while lines 8-16 create the new generation.

```

1. GRHEA( $n, l$ )
2.   forall  $i \in 1..PopulationSize$ 
3.      $\Sigma \leftarrow \Sigma \cup \{\text{RANDOMCONFIGURATION}(n, l)\}$ ;
4.    $g \leftarrow 0$ ;
5.   while  $g \leq maxGen \ \& \ v(\Sigma) > 0$  do
6.      $i \leftarrow 0$ ;
7.      $\Sigma^+ \leftarrow \emptyset$ ;
8.     while  $i \leq populationSize$  do
9.       select  $(\sigma_1, \sigma_2) \in \Sigma$ ;
10.      with probability  $P_c$ 
11.         $(\sigma_1, \sigma_2) \leftarrow \text{crossover}(\sigma_1, \sigma_2)$ ;
12.      with probability  $P_m$ 
13.         $\sigma_1 \leftarrow \text{GRLS}(\sigma_1)$ ;
14.         $\sigma_2 \leftarrow \text{GRLS}(\sigma_2)$ ;
15.       $\Sigma^+ \leftarrow \Sigma^+ \cup \{\sigma_1, \sigma_2\}$ ;
16.       $i \leftarrow i + 2$ ;
17.     $\Sigma \leftarrow \Sigma^+$ ;
18.     $g \leftarrow g + 1$ ;
19.    if  $diversity(\Sigma) < \Upsilon$ 
20.      forall  $i \in 1..PopulationSize$ 
21.         $\Sigma \leftarrow \Sigma \cup \{\text{RANDOMCONFIGURATION}(n, l)\}$ ;

```

Figure 2: Algorithm GRHEA for Finding Golomb Rulers

The new rulers are generated by selecting the parents in line 9, applying a crossover with probability P_c (lines 10-11), and applying a mutation with probability P_m (lines 12-14). Note that the function $\text{GRHEA}(\sigma)$ denotes the execution of GRLS starting from ruler σ . The new rulers are added to the new population in line 15. The new population becomes the current population in line 17. If the new population is not diverse enough (line 19), it is reinitialized from scratch (lines 20-21).

Experimental Results Table 1 reports the experimental results for algorithm GRHEA. Algorithm GRHEA was run 50 times for each ruler with a population size of 50 and with a maximum of 10,000 iterations for the local search. The crossover and mutation probabilities were both set to 0.6 and the diversity parameter Υ is set to 5. These parameters were determined from a limited number of experiments and can certainly be tuned for specific instances. Only results with roulette selection are reported. The results of random selection are relatively close, but near-optimal results only use roulette selection.

4 Near Optimal Golomb Rulers

The algorithms described so far compute Golomb rulers of specified lengths. This section discusses how to generalize them to find near-optimal Golomb rulers.

4.1 The Difficulty

Consider first the problem of generalizing the tabu-search algorithm for finding near-optimal Golomb rulers. A natural approach is to solve a sequence of feasibility problems.

```

1. GROHEA( $n, u$ )
2.    $\sigma \leftarrow \text{GRGHEA}(n, u)$ ;
3.   while  $v(\sigma) = 0$  do
4.      $\sigma^* \leftarrow \sigma$ ;
5.      $\sigma \leftarrow \text{GRGHEA}(n, \text{LENGTH}(\sigma^*)-1)$ ;
6.   return  $\sigma^*$ ;

```

Figure 3: Algorithm GROHEA for Near-Optimal Rulers

Starting from an upper bound l on the optimal length of the ruler, the algorithm then searches for rulers of length $l, l-1, \dots$ until no solution can be found. This approach, although conceptually simple, performs poorly. Indeed, it essentially solves a sequence of mostly unrelated problems, since no information is reused across the searches and, in addition, the search for a ruler of length l is not necessarily simpler than the search for a ruler of smaller length.

A second approach consists of integrating the ruler length as part of the objective function and to consider the last mark m_n as a decision variable. The objective function now combines constraint violations and the ruler length in order to guide the search toward optimal rulers. The violations and the length can be combined in different fashions. However, preliminary experimental results with this approach were not encouraging, although there may exist effective ways to combine these two conflicting objectives effectively for tabu-search or other meta-heuristics.

4.2 Generalizing the Hybrid Evolutionary Algorithm

Interestingly, the HE algorithm can be generalized to produce an indirect, but effective, approach for finding near-optimal Golomb rulers in reasonable time. The approach consists again of solving a sequence of feasibility problems, starting from an upper bound l and producing a sequence of rulers of length $l_1 > l_2 > \dots > l_i > \dots$. The key idea however is not to fix the length of the ruler in the HE algorithm. More precisely, the new HE (GRGHEA) algorithm considers the last mark m_n as a decision variable whose value is at most l , where l is the best available upper bound. The initial population consists of random rulers whose lengths are at most l , but are likely to be shorter. Crossover operations proceed as before. Mutations are again performed by the local search algorithm which still minimizes the number of violations but now considers the last mark as a decision variable. This algorithm differs from the previous one only in lines 13 and 14 (figure 2), where instead of using the GRLS, it would make use of a slightly modified procedure which will take into account the last mark of the ruler; this translates to the fact that $\sigma(m_n)$ is now not fixed (see remark in equation 5). Note that the length of the ruler is not incorporated in the objective function which focuses exclusively on feasibility.

The generalized HE algorithm GROHEA for finding near-optimal rulers is depicted in Figure 3. Given an upper bound u on the length of an n -mark ruler, the algorithm first searches for a ruler of length at most u (line 2). It then performs a number of iterations, each of which producing rulers of smaller length (lines 3-5), until no feasible solu-

# marks	CPU Time(secs)		Local Moves		Failures			CLS	
	avg	mdn	avg	mdn	%F	MaxGen	time Uns.	Backtracks	CPU Time
5	0.0	0.0	3.46	1	0	10	-	15	0.0
6	0.0	0.0	8.78	5	0	10	-	24	0.0
7	0.0	0	42.44	21	0	10	-	145	0.0
8	0.03	0.02	1125.54	564	0	10	-	5114	0.08
9	0.24	0.19	5711.32	4339.5	0	10	-	23118	0.47
10	3.49	2.29	5674.5	37479.5	0	10	-	74860	1.87
11	8.15	5.86	84606.2	60836.5	0	10	-	269905	8.16
12	199.45	166.75	1531640.67	1288230.5	1	20	2411.1	2005597	72.2
13	1071.74	959.55	5655670.67	4969063	1	50	990.36	20360198	860
14	1013.2	3861.69	3939817.5	14965000	98	50	3860.93		

Table 1: Experimental Results of GRHEA for Rulers from 5 to 14 Marks.

tion can be found for a specified length. The main step is in line 5: It uses the HE algorithm GRGHEA for finding a ruler of length smaller than $\text{LENGTH}(\sigma^*)$, where σ^* is the smallest ruler found so far and $\text{LENGTH}(\sigma^*)$ is simply the value $\sigma^*(m_n)$ of the last mark. Note also that algorithm GRGHEA is the HE algorithm GRHEA(n, l) presented earlier, except that the last mark is now a decision variable and the initial population are rulers whose length is at most l but may be shorter.

Algorithm GROHEA is best viewed as solving a sequence of feasibility problems to find rulers of decreasing lengths. However, algorithm GROHEA does not artificially constrain the ruler length. Instead, the search is directed by constraint violations and the length of the ruler, i.e., the value of the last mark, is modified appropriate to minimize violations.

4.3 Experimental Results

Table 2 reports the experimental results for algorithm GROHEA and compare them with the HGRASP algorithm of [Cotta & Fernández 2004]. All experiments use a roulette wheel selection and are based on the following settings. The maximum number of iterations for the tabu search is 10,000, the size of the population is 50, the probabilities P_c and P_m are both 0.6, and Υ is 5. For a n -mark ruler, the algorithm uses the optimal length of an $n + 1$ -mark ruler as initial upper bound and is iterated until no improved ruler is found for two successive generations, except for $n = 16$ where we use three generations. The GROHEA is run 30 times for each ruler (like the HGRASP in [Cotta & Fernández 2004]). Finally, we also let algorithm GROHEA without time/generation limits to determine whether it can find optimal rulers (these results are for a small number of runs). Both algorithms were run on similar machines.

The table reports the best and median lengths for rulers with 11 to 16 marks found by algorithms HGRASP and GROHEA within their time limits (algorithm GROHEA easily finds optimal rulers for smaller lengths). It also reports the average times of both algorithms in minutes. In addition, for algorithm GROHEA, the table also gives the time to find the last solution. The last column reports the time of GROHEA to find optimal rulers.

The results are particularly impressive. First observe that GROHEA systematically finds optimal rulers up to 11 marks very quickly. Algorithm HGRASP does not find optimal rulers systematically even for 10 marks and never finds optimal rulers for 11 marks. Algorithm GROHEA also finds optimal rulers for 12 and 13 marks in less than two minutes and for 14 marks in about 40 minutes. Algorithm GROHEA also improves the near-optimal solutions significantly. For 14 marks, the best solutions of GROHEA are with 3.1% of the optimal rulers (instead of 6.3% for HGRASP) in about 6 minutes. They are with 4.6% and 5.6% for 15 and 16 marks in about 9 and 13 minutes. These represent improvements ranging from 1.4% to 3.2% compared to HGRASP. Similar results are obtained for median values as well.

A fundamental benefit of GROHEA is its ability to improve its solutions over time, which does not seem to be the case of prior generic and/or hybrid evolutionary algorithms. Contrary to GROHEA, earlier algorithms were not able to find optimal solutions for 13 and 14 marks. Algorithm GROHEA also finds a solution of length 153 in about an hour on 15 marks (151 is the optimal length), showing that better solutions can be found when the algorithm is given more time. This is particularly interesting given the natural modeling and conceptual simplicity of the algorithm.

It is also important to stress that these results were obtained without tuning of the parameters. In particular, larger instances are likely to benefit from longer tabu searches and, possibly, more sophisticated crossovers. But the results clearly indicate the potential of hybrid evolutionary algorithms for finding near-optimal rulers.

5 Conclusion

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.

This paper presented the hybrid evolutionary algorithm GROHEA to find near-optimal Golomb rulers in reasonable time. The algorithm is conceptual simple and uses a natural modeling. It incorporates a tabu-search algorithm for

#marks	Opt	HGRASP			GROHEA				
		Best	Median	Time	Best	Median	Time	Last(time)	Opt(time)
11	72	74(2.8)	74(2.8)	1.5	72	72	0.3	0.3	0.1
12	85	94(10.6)	95(11.8)	2.4	85	91(7.1)	2.3	1	1.8
13	106	111(4.7)	114(7.5)	3.6	106	112(5.6)	3.9	2	1.7
14	127	135(6.3)	139(9.4)	5.3	131(3.1)	136(7.1)	6	3.2	40.7
15	151	162(7.3)	169(11.9)	7.6	158(4.6)	164(8.6)	8.7	4.7	-
16	177	189(6.8)	197(11.3)	11.3	187(5.6)	195(10.2)	13.4	5.9	-

Table 2: Experimental Results for the GROHEA Algorithm

mutation and a one-point crossover to cross two rulers. It optimizes the length of the rulers indirectly by solving a sequence of feasibility problems.

Algorithm GROHEA significantly outperforms earlier (hybrid) genetic algorithms for finding Golomb rulers and compares favorably to hybridizations of local search and constraint programming. In particular, it finds optimal rulers for up to 14 marks (earlier genetic algorithms did not find optimal ruler for 12 marks in reasonable time). It obtains near-optimal solutions quickly for up to 16 marks, improving earlier algorithms by about 2% on the larger instances (and more on the smaller instances). One of the main benefits of GROHEA is its ability to find optimal solutions even for large rulers, which does not seem to be the case of earlier evolutionary algorithms.

Overall, these results demonstrates the potential of hybrid evolutionary algorithms for finding Golomb rulers. They also leave many interesting open issues. First, it would be interesting to determine whether GROHEA can find optimal solutions for m -rulers with $m \geq 15$. Second, it is important to study more involved local search (using more sophisticated neighborhoods) and crossover operators to increase diversity and reach better solutions faster. Finally, it is worth integrating some ideas of earlier algorithms into GROHEA, since some of the techniques are orthogonal and produces good upper bounds very quickly.

Acknowledgements

We would like to thank the reviewers for many useful comments on our paper.

Bibliography

- [Bean 1994] Bean, J. 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing* 6:154–160.
- [Bloom, & Golomb 1977] Bloom, G., Golomb, S. 1977. Applications of numbered undirected graphs. In Proceedings of *IEEE*, volume 65, 562–570.
- [Blum, Biraud, & Ribes 1974] Blum, E., Biraud, F., Ribes, J. 1974. On optimal synthetic linear arrays with applications to radioastronomy. In *IEEE Transactions on Antennas and Propagation*, 108–109.
- [Cotta & Fernández 2004] Cotta, C., and Fernández, A. 2004. A hybrid grasp - evolutionary algorithm approach

to golomb ruler search. In *Parallel Problem Solving from Nature VIII*. Lecture Notes in Computer Science 3242.

- [Dolas, Rankin, & McCracken 1998] Dolas, A.; Rankin, W.; and McCracken, D. 1998. New Algorithms for Golomb Ruler Derivation and Proof of the 19 Mark Ruler. *IEEE Transactions on Information Theory* 44:379–382.
- [Feeney 2003] Feeney, B. 2003. Determining Optimum and Near-Optimum Golomb Rulers using Genetic Algorithms. ScM Thesis, University College Cork.
- [Hayes 1998] Hayes, B. 1998. Collective wisdom. In *American Scientist* 86, 118–122.
- [Klove 1989] Klove, T. 1989. Bounds and construction for difference triangle sets. In *IEEE Transactions on Information Theory*, IT-36.
- [Pereira, Tavares, & Costa 2003] Pereira, F.; Tavares, J.; and Costa, E. 2003. Golomb Rulers: the Advantage of Evolution. In *Eleventh Portuguese Conference on Artificial Intelligence*, volume 2902, 29–42. Springer-Verlag.
- [Prestwich 2001] Prestwich, S. 2001. Trading Completeness for Scalability: Hybrid search for Cliques and Rulers. In *CPAIOR-01*, 159–174.
- [Smith, Stergiou, & Walsh 1999] Smith, B.; Stergiou, K.; and Walsh, T. 1999. Modelling the Golomb Ruler Problem. In *IJCAI'99 Workshop on Non-Binary Constraints*.
- [Soliday, Homaifar, & Leby 1995] Soliday, S.; Homaifar, A.; and Leby, G. 1995. Genetic Algorithm Approach to the Search for Golomb Rulers. In *Sixth International Conference on Genetic Algorithms*, 528–535. Pittsburgh, PA, USA: Morgan Kaufmann.