Constraint-Based Local Search for Golomb Rulers

M.M. Alam Polash[1](✉), M.A. Hakim Newton[1], and Abdul Sattar[1,2]

[1] Institute for Integrated and Intelligent Systems, Griffith University, Nathan, Australia
mdmasbaul@gmail.com, {mahakim.newton,a.sattar}@griffith.edu.au
[2] Queensland Research Lab, National ICT Australia, Sydney, Australia

Abstract. This paper presents a constraint-based local search algorithm to find an optimal Golomb ruler of a specified order. While the state-of-the-art search algorithms for Golomb rulers hybridise a range of sophisticated techniques, our algorithm relies on simple tabu meta-heuristics and constraint-driven variable selection heuristics. Given a reasonable time limit, our algorithm effectively finds 16-mark optimal rulers with success rate 60 % and 17-mark rulers with 6 % near-optimality.

Keywords: Golomb ruler · Constraints · Local search · Tabu meta-heuristics

1 Introduction

Golomb rulers were described in 1977 by S.W. Golomb [4] although such concept was already conceived in 1953 by W.C. Babcock [3]. A *Golomb ruler* of *order* $m > 0$ and *length* n is a sequence of m integers called *marks* $0 = x_1 < x_2 < \cdots < x_m = n$ such that each $x_j - x_i$ is unique for $1 \leq i < j \leq m$. A Golomb ruler of order m is *optimal* if n is the minimum possible integer. Golomb rulers have a wide variety of applications that include x-ray crystallography [4], radio astronomy [5], information theory [18] and pulse phase modulation [17].

Finding an Optimal Golomb Ruler (OGR) is an extremely difficult task. It takes 36200 CPU hours to find a 19-mark Golomb ruler on a Sun SPARC workstation using an exhaustive parallel search algorithm [8]. OGR is a combinatorial problem whose bounds grow geometrically with respect to the solution size [19]. The major limitation is that each new ruler to be discovered is, by necessity, larger than its predecessor. However, the search space is bounded and, therefore, solvable [13]. Also, for a given order, more than one OGR may exist.

To solve this highly combinatorial problem, a number of approaches have already been developed before. However, the current state-of-the-art results come from a sophisticated hybrid method [7] that combines ideas from greedy randomised adaptive search procedure (GRASP), scatter search (SS), tabu search

L. Michel (Ed.): CPAIOR 2015, LNCS 9075, pp. 322–331, 2015.
DOI: 10.1007/978-3-319-18008-3_22

(TS), clustering techniques and constraint programming. The hybrid algorithm found 16-mark OGRs with success rates 5-10%. Nevertheless, an analysis of the fitness landscape of OGR presented in [7] shows that high irregularities in the neighbourhood structure introduce a drift force towards low-fitness regions of the search space. For higher order rulers, search algorithms thus quickly reach a near-optimal value and then stagnate around it, apparently causing a *cycling problem* and making the search space less accessible. A restarting mechanism is therefore needed at that stage.

In this paper, we present a constraint-based local search approach to find Golomb rulers. Our algorithm takes m and n as input and finds the m integers of the Golomb ruler. For OGRs, we assume x_m to be equal to the optimal n for order m. For near-optimal rulers, we assume x_m to be less than or equal to a given n. Instead of a sophisticated hybridisation of a range of techniques, we rather rely on simple tabu meta-heuristics and constraint-driven variable selection heuristics. Besides the traditional way of enforcing tabu on recently modified variables for a given number of iterations, we use a special type of tabu called *configuration checking* (CC) [6]. The CC strategy for OGR prevents a variable (i.e. a mark) from being selected if it is fully confined by neighbouring variables. The use of CC effectively reduces the number of restarts required during search and thus mitigates the cycling problem of local search for finding an OGR. Experimental results show that within a reasonable time limit, our algorithm effectively achieves significantly high success rate of 60% in finding OGRs of order 16 and about 6% near-optimal rulers of order 17.

The rest of the paper is organised as follows: Section 2 explores related work; Section 3 describes our approach; Section 4 presents the experimental results; and finally, Section 5 draws our conclusions.

2 Related Work

Various techniques have been applied so far to find Golomb rulers. Scientific American algorithm, token passing algorithm, shift algorithm are described and compared in [16]. Geometry tools such as projectile plane construction and affine plane construction are used in a non-systematic method in [10]. A systematic branch and bound algorithm along with the Depth First Search (i.e backtracking algorithm) is proposed in [19]. A genetic algorithm is proposed in [21]. Constraint programming techniques are used in [20]. A combination of constraint programming and sophisticated lower bounds for Golomb rulers are used in [12]. A hybrid of local search and constraint programming is proposed in [15].

Three algorithms, namely a genetic algorithm on its own, then with local search and Baldwinian learning, and with local search and Lamarckian learning are studied in [11]; the best results have the distance between 6.8 and 20.3% from the optimum. A simple hybrid evolutionary algorithm (called GRHEA) is presented in [9] to find an OGR of a specified length. Also, an indirect but effective approach (called GROHEA) is proposed to find near-optimal Golomb rulers. For a given order m, GROHEA starts from an upper bound of n and if a Golomb ruler of length n is found, it then tries to find another one with length $n - 1$. GROHEA systematically finds optimal rulers for up to 11 marks very quickly. It also finds optimal rulers for 12 and 13 marks in less than two minutes, and for 14 marks in about 40 minutes. For 15 and 16 marks, the best solutions of GROHEA are within 4.6% and 5.6% of the optimal rulers.

2.1 A Recent Hybrid Local Search Algorithm

This algorithm [7] combines the greedy randomised adaptive search procedure (GRASP), evolutionary algorithms (EA), scatter search (SS), tabu search (TS), clustering techniques, and constraint programming (CP) to find optimal or near-optimal Golomb rulers. To find OGRs, this algorithm first uses an indirect approach, which incorporates GRASP. However, one major problem of the basic GRASP procedure is that it relies on certain parameter values to select an attribute value from the Ranked Candidate List (RCL). The choice of the parameter value often hinders to find high-quality solutions. GRASP is therefore combined with EA in HEAGRASP so that different parameter values can be used in each application of the ruler construction phase. The plain GRASP and HEAGRASP can find OGRs up to 9 and 10 marks respectively.

The work in [7] then proposes a scatter search (SS) that is basically a memetic algorithm. SS uses an indirect approach in the initialisation and restarting phase (ideas borrowed from HEAGRASP) and a direct approach in the local improvement and recombination phase. More specifically, the TS is used as the local improvement method. SS can find OGRs for up to 15 marks and computes high quality near-optimal solutions for 16 (i.e. 1.1% from the optimum). SS is further enhanced by using a complete search in recombination of individuals and a clustering procedure to achieve higher degree of diversity. As a result, 16-mark OGRs are found with success rates 5-10%.

2.2 A Recent Hybrid Genetic Algorithm

Recently, a hybrid genetic algorithm is presented in [2] to find optimal or near-optimal Golomb rulers. This approach has been able to obtain OGRs for up to 16 marks at the expense of an important execution time. For instance, around 5 hours for 11-mark, 8 hours for 12-mark, and 11 hours for 13-mark ruler. It is also able to find near-optimal rulers for 20 and 23 marks using enormous time. The parallel implementation of this algorithm can be found in [1].

3 Our Approach

Our approach is based on a constraint-based local search algorithm. It is a simple but effective algorithm to find an OGR of a specified order. Given the optimal length n of an m-mark ruler, it searches for a ruler that satisfies the criteria of an optimal one. Note that the first and last marks are fixed to 0 and n respectively. For a near-optimal Golomb ruler, the last mark remains flexible to take a value less than or equal to n', where n' is the optimal length of a $(m+1)$-mark ruler. To overcome the cycling problem of local search, we use the tabu mechanism and the configuration checking techniques [6]. Nevertheless, below we provide a detailed description of our search algorithm.

3.1 Problem Model

We represent a Golomb ruler of order m and length n by using m variables $x_1, x_2, \cdots, x_m$. Without loss of generality, we fix the value of x_1 at 0 and the value of x_m at n. Initially, the domain of all other marks is defined $x_i \in [1, n-1]$. However, we assume the ordering $x_1 < x_2 < \cdots x_m$. As the search progress, the domain of a mark $x_i (1 < i < m)$ is thus dynamically restricted by the values of its neighbours. Thus, $x_i \in [x_{i-1}+1, x_{i+1}-1]$ for each $1 < i < m$.

To define the constraint model, for each $i > j$, we first calculate the difference expression $d_{ij} = x_i - x_j$. The value of d_{ij} is the *distance* between x_i and x_j. We then define an *alldifferent* constraint on the d_{ij}s. To guide the search, the constraint violation metric is calculated as in [9]. Given the current solution R i.e. the values of all x_is, the violation $V_R(d)$ of a distance d is the number of times distance d appears between two marks beyond its allowed occurrences.

$$V_R(d) = max(0, \#\{d_{ij} = d | 1 \le j < i \le m\} - 1) \tag{1}$$

The violation *V(R)* of the current ruler R is simply the sum of $V_R(d)$s:

$$V(R) = \sum_{d=1}^{n} V_R(d) \tag{2}$$

Obviously, a ruler R with $V(R) = 0$ is a solution to the Golomb ruler problem. In each iteration, our algorithm will try to minimise the value of $V(R)$.

To define a variable selection heuristic, we further define the violation metrics for each difference d_{ij} and for each variable $x_i (1 < i < m)$. The violation for each distance d_{ij} will be the violation of its distance value.

$$V_R(d_{ij}) = V_R(d) \tag{3}$$

where $d_{ij} = d$ in R. The violation of a variable is calculated by summing the violations of the distances that depend on that variable.

$$V_R(x_i) = \sum_{k=1}^{i-1} V_R(d_{ik}) + \sum_{k=i+1}^{m} V_R(d_{ki}) \tag{4}$$

During search, we mainly follow max/min style search. At each iteration, a variable $x_i (1 < i < m)$ having the maximum $V_R(x_i)$ is selected first and then a value $v \in [x_{i-1}+1, x_{i+1}-1]$ that minimises V_R is selected for x_i.

3.2 Avoiding the Cycling Problem

The cycling problem in local search has been typically tackled by the tabu mechanism. Besides using the tabu mechanism, in this paper, we use another recently emerging strategy called *configuration checking* (CC) [6].

Tabu Mechanism. By maintaining a parameter called tabu tenure, the tabu mechanism prevents the local search to immediately return to a previously visited candidate solution. In our algorithm, we tabu the variable selected in the last iteration and we tabu it for the tabu-tenure period.

Configuration Checking. The CC [6] strategy reduces the cycling problem by checking the circumstance information. The core idea is: A variable's value should not change until at least one of its neighbouring variables has a new value. Since in our algorithm, a variable's value depends on its neighbour as we enforce $x_i \in [x_{i-1}+1, x_{i+1}-1]$ during search, CC is relevant here along with the tabu mechanism. However, we use CC in our algorithm in a special case when a variable's range has nothing but its own current value. In this case, the variable is locked for any future changes until any of its neighbouring variables have changed.

3.3 Search Algorithm

Our constraint-based local search algorithm to find Golomb rulers is shown in Algorithm 1. The core of the algorithm is in *Lines* 4–19 where local moves are performed for a number of iterations or until a solution is found. The unlocked variable with the highest number of violations is selected in *Line* 8 and a value for that variable is selected in *Line* 9 such that the number of violations decreases after assigning the value to the variable. The new ruler is generated in *Line* 13 and the tabu is applied on the variable in *Line* 14. Note that the tabu tenure tt is normally within 3 to 5. *Line* 11 and 15 implement the idea of CC. CC locks a variable whenever its domain contains no value except the current one. When a new value is set into a variable, CC unlocks the neighbours of that variable provided they are already locked. *Lines* 16–19 update the best violation metric and plateau size depending on the progress of violation metrics. *Lines* 5–6 restart the search when the current plateau size exceeds a given limit.

Initialisation and Restarting Mechanism. The initial ruler is generated by selecting random values from the initial domain of each mark. Special care is taken so that no two marks have the same value. The marks are then sorted to obtain an ordered ruler. When our search algorithm gets stuck showing no progress, we just use the initialisation procedure to restart the search from scratch. We detect the stagnation situation when the global best violation seen so far does not change for a given number of iterations.

Algorithm 1. Constraint-Based Local Search for Golomb Rulers

```
1  Parameters: order $m$, length $n$, tabu tenure $tt$
2  Generate an initial solution $R$ using an initialisation procedure
3  plateauSize = 0, iteration = 0, bestViolation = $V(R)$
4  while $++iteration \leq maxIteration$ and $bestViolation > 0$ do
5      if $plateauSize > maxPlateauSize$ then
6          Restart from scratch and set plateauSize = 0
7      else
8          Select the unlocked variable $x_k (1 < k < m)$ with the highest $V_R(x_k)$
9          Select a value $v \in [x_{k-1} + 1, x_{k+1} - 1]$ such that $V(R)$ is minimised
10         if $v$ is $x_k$'s current value then
11             lock $x_k$ to stop its future changes //part of CC
12         else
13             set $x_k = v$ in the current ruler $R$
14             apply tabu on $x_k$ for the specified tabu tenure $tt$
15             unlock $x_{k-1}$ and $x_{k+1}$ if they are locked //part of CC
16     if $currentViolation < bestViolation$ then
17         bestViolation = currentViolation, plateauSize = 0
18     else if $currentViolation == bestViolation$ then
19         plateauSize++
```

4 Experiments and Analyses

We implemented our algorithm using C++ and on top of the constraint-based local search system, Kangaroo [14]. The functions and the constraints are defined using invariants in Kangaroo. Invariants are special constructs that are defined by using mathematical operators over the variables. While propagation of violations, simulation of moves, execution and related calculations are performed incrementally by Kangaroo, we mainly focus on the search algorithms.

We ran our experiments on High Performance Computing Cluster Gowonda provided by Griffith University. Each node of the cluster is equipped with Intel Xeon CPU E5-2650 processors @2.60 GHz, FDR 4x infiniBand Interconnect, having system peak performance 18949.2 Gflops. Our search algorithm is run for 25 times with timeout 48 hours for each given order of the Golomb ruler. The tabu tenure is between 3 and 5. For a given order m and its optimal length n, we run our algorithm to find an OGR first. If an OGR is not found, then we consider finding a near-optimal Golomb ruler with an increased ruler length. Our experimental results are shown in Table 1 and Figure 1. Left most two columns in the Table 1(a) show the orders 11–16 and the optimal lengths of the Golomb rulers. Moreover, the columns under "TabuAndCC" show our final results.

4.1 Effectiveness of CC

To investigate the effectiveness of CC, we run a version of our algorithm that does not use the CC strategy. These results are shown in Table 1(a) in the columns under the header "TabuNoCC". Also, the charts in Figure 1 show how the two versions' success rate differ when various timeout limits are assumed. Overall, we observe the TabuAndCC version, our final algorithm, significantly outperforms the TabuNoCC version in obtaining higher order OGRs, in success rates, and in running times. To analyse further, in Table 1(b), we show the number of restarts required by the TabuAndCC and TabuNoCC versions of our algorithm. As we can see, the number of restarts required for the TabuAndCC version is very small compared to the TabuNoCC version. It exhibits that the use of CC effectively reduces the occurrence of stagnation in the search.

Table 1. (a) Performance of our algorithm when compared to GRHEA[9]. Time statistics for GRHEA is collected from the published article while our algorithms are run on our computers. (b) Average numbers of restarts required during search (c) Approximate average numbers of candidate solutions explored/evaluated during search.

Num	Opt	**TabuAndCC**		**TabuNoCC**		**GRHEA[9]**	
of	GR	*Succ*	*Median*	*Succ*	*Median*	*Succ*	*Median*
Marks	Len	*Rate*	*Time*	*Rate*	*Time*	*Rate*	*Time*
11	72	100	1.23 S	100	7.96 S	100	5.86 S
12	85	100	28.09 S	100	1.65 M	99	2.78 M
13	106	100	6.47 M	100	11.39 M	99	15.99 M
14	127	100	2.76 H	96	5.64 H	2	1.07 H
15	151	84	3.26 H	76	22.63 H		
16	177	60	14.83 H				

(a)

Marks	TabuAndCC	TabuNoCC
11	1.2	366.2
12	64.32	18214.12
13	234.08	77233.4

(b)

Marks	TabuAndCC	GRHEA[9]
11	1×10^6	10×10^6
12	4.5×10^7	23×10^7
13	5.1×10^8	11×10^8

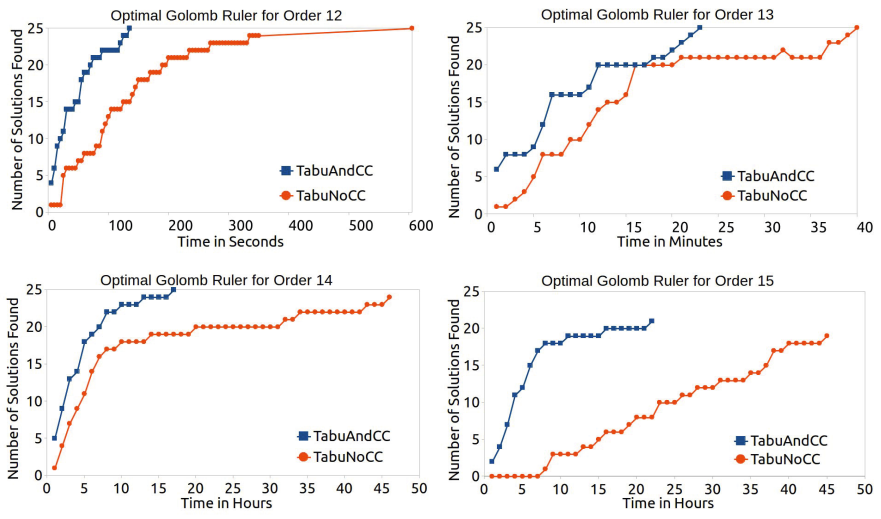
(c)

4.2 Optimal Golomb Rulers

We compare our algorithm with other state-of-the-art algorithms for Golomb rulers. As we see in Table 1(a), GRHEA[9] can solve up to order 14 but with success rate for 14 being 2%. HybridGA[2] claims to have solved up to 16 but success rates are not mentioned and run-times are either enormous or not reported. The Hybrid Local Search [7] states to have consistently found OGRs for up to 14 marks and for 16 marks with 5-10% success rate[1]. The OGR problem gets extremely hard from order 16 onward [7]. Notice that our final algorithm (Columns "TabuAndCC") obtains significantly better results with 100%, 84% and 60% success rates for 14, 15 and 16-mark rulers respectively.

We compare the search effort behind the performance of our algorithm and those in [7,9] on the average number of candidate rulers explored and evaluated. Comparison on execution time is not possible because experiments in [7,9] were run on a number of different machines[1]. Note that while the algorithms in [7,9] are variants of memetic algorithms (in other words, local searh is used as a mutation operator in a genetic algorithm), our algorithm is just a constraint-based local search algorithm. So for a fair comparison between these different types of

[1] E-mail communication with Antonio J. Fernandez, one of the authors of [7].



TabuAndCC for a 16-mark ruler																										
Time in Hours	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	22	23	24	25
Solutions Found	1	2	2	3	4	5	5	6	6	6	6	6	6	7	8	8	10	13	13	13	13	13	13	14	14	15

Fig. 1. Number of times the optimal ruler is found when 25 attempts are made within given time limits. The largest timeout was 48 hours for all runs. The times in x-axis are in seconds for order 12, in minutes for 13, and in hours for 14 and 15. The times in the first row of the table is also in hours for the ruler of order 16.

algorithm, particularly when time comparison is not possible, we consider the average numbers of rulers explored and evaluated by each algorithm to be an appropriate criterion; similar notions were used in [7,9].

To obtain the maximum numbers of rulers explored or evaluated by GRHEA [9], we take the maximum number of generations (50), the population size (50), the probability to call the LS procedure for each individual (0.6), the number of iterations in the LS (10000). We get the numbers of LS iterations altogether to be $50 \times 50 \times 0.6 \times 10000 = 15 \times 10^6$. In each LS iteration, GRHEA considers each variable and its value from the range bounded by the two neighbour variables' values. In GRHEA's model, a Golomb ruler with m mark and length n has thus $2(n - m + 1)$ neighbours. So $(n - m + 1) \times 30 \times 10^6$ gives the maximum numbers of rulers explored or evaluated. However, instead of 15×10^6, Table 1(c) uses the average number of local moves made by GRHEA as reported in [9].

For our algorithm, we just take the average numbers of iterations and then consider the fact that in each iteration, unlike in GRHEA, only one variable is selected heuristically and then like GRHEA, the value of the variable is selected only from those bounded by the neighbouring two variables' values. So the number of neighbours on an average is $2(n - m + 1)/(m - 1)$. Table 1(c) shows that our algorithm puts significantly less effort in search than GRHEA but obtains significantly better performance. The average numbers of explored candidate solutions by Hybrid Local Search [7] are similar to GRHEA because in [7] these algorithms are compared giving the same effort in search.

4.3 Near-Optimal Golomb Rulers

As noted before, for OGRs of order m, in our algorithm, we set $x_m = n$ where n is the optimal length for order m. For near-optimal Golomb rulers of order m, we run the same algorithm but with $x_m \leq n'$ where n' is the optimal length for a Golomb ruler of order $m+1$. Our algorithm that uses "TabuAndCC" can find OGRs of order 16 but not of 17. So for the time being, we further run it to find near-optimal Golomb rulers of order 17. We do not run our algorithm that uses "TabuNoCC" for order 17 because it could not find OGRs even for 16. *Table* 2 shows that our "TabuAndCC" algorithm finds near-optimal rulers for order 17 with a success rate of 100% and the best solutions found have lengths within 6.03% of that of the optimal rulers.

Table 2. Performance of our algorithm in finding near-optimal Golomb ruler

Mark	n	n'	Success Rate	Best Length	Mean Length	Median Length	Mean Time	Median Time
17	199	216	100	211	214	215	6.97 Hours	5.88 Hours

5 Conclusion

We have presented a constraint-based local search algorithm that takes the number of marks and length of an optimal Golomb ruler as input and finds the positions of the marks. Our algorithm is simple, but given a reasonable time limit, it effectively finds 16-mark optimal rulers with 60% success rate and about 6% near-optimal 17-mark rulers.

References

1. Ayari, N., Jemai, A.: Parallel hybrid evolutionary search for Golomb ruler problem. In: International Conference on Metaheuristics and Nature Inspired Computing (META) (2010)
2. Ayari, N., Luong, T., Jemai, A.: A hybrid genetic algorithm for golomb ruler problem. In: IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), pp. 1–4 (2010)
3. Babcock, W.: Intermodulation interface in radio systems. Bell Systems Technical Journal, 63–73 (1953)
4. Bloom, G., Golomb, S.: Application of numbered undirected graphs. In: Proceedings of the IEEE, vol. 65(4), pp. 562–570 (1977)

5. Blum, E., Biraud, F., Ribes, J.: On optimal synthetic linear arrays with applications to radioastronomy. IEEE Transactions on Anntenas and Propagation (1974)
6. Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence **175**(9–10), 1672–1696 (2011)
7. Cotta, C., Dotu, I., Fernandez, A.J., Hentenryck, P.V.: Local search based hybrid algorithm for finding Golomb rulers. Contraints, 263–291 (2007)
8. Dollas, A., Rankin, W., McCracken, D.: A new algorithm for Golomb ruler derivation and proof of the 19-mark ruler. IEEE Transactions on Information Theory, 379–386 (1998)
9. Dotu, I., Hentenryck, P.V.: A simple hybrid evolutionary algorithm for finding Golomb rulers. IEEE Congress on Evolutionary Computation (2005)
10. Drakakis, K.: A review of the available construction methods for Golomb rulers. Advances in Mathematics of Communications **3**(3), 235–250 (2009)
11. Feeny, B.: Determining optimum and near-optimum Golomb rulers using genetic algorithms. Master's thesis, Computer Science, University College Cork, October 2003
12. Galinier, P., Jaumard, B., Morales, R., Pesant, G.: A constraint-based approach to the golomb ruler problem. In: 3rd International Workshop on CPAIOR (2001)
13. Klove, T.: Bounds and construction for difference triangle sets. IEEE Transactions on Information Theory, 879–886 (1989)
14. Newton, M.A.H., Pham, D.N., Sattar, A., Maher, M.: Kangaroo: an efficient constraint-based local search system using lazy propagation. In: Lee, J. (ed.) CP 2011. LNCS, vol. 6876, pp. 645–659. Springer, Heidelberg (2011)
15. Prestwich, S.: Trading completeness for scalability: hybrid search for cliques and rulers. In: 3rd International workshop on CPAIOR, pp. 159–174 (2001)
16. Rankin, W.: Optimal Golomb rulers: An exhaustive parallel search implementation. Master's thesis, Duke University Electrical Engineering Dept., Durham, NC, December 1993
17. Robbins, J., Gagliardi, R., Taylor, H.: Acquisition sequences in PPM communications. IEEE Transactions on Information Theory, 738–744 (1987)
18. Robinson, J., Bernstein, A.: A class of binary recurrent codes with limited error propagation. IEEE Transactions on Information Theory, 106–113 (1967)
19. Shearer, J.: Some new optimum Golomb rulers. IEEE Transactions on Information Theory, 183–184 (1990)
20. Smith, B., Walsh, T.: Modelling the golomb ruler problem. In: Workshop on Non-Binary Constraints (IJCAI), Stockholm (1999)
21. Soliday, S., Homaifar, A., Lebby, G.: Genetic algorithm approach to the search for golomb rulers. In: Eshelman, L.J. (ed.) 6th International Conference on Genetic Algorithm (ICGA 1995), pp. 528–535. Morgan Kaufmann, Pittsburg (1995)