Carnegie Mellon University
Research Showcase @ CMU

Tepper School of Business

Spring 2013

A Lagrangian Relaxation for Golomb Rulers

Marla R. Slusky
Carnegie Mellon University, mslusky@andrew.cmu.edu

Willem-Jan van Hoeve
Carnegie Mellon University, vanhoeve@andrew.cmu.edu

Follow this and additional works at: http://repository.cmu.edu/tepper

 Part of the Business Commons

Published In
Proceedings of CPAIOR.

This Conference Proceeding is brought to you for free and open access by Research Showcase @ CMU. It has been accepted for inclusion in Tepper School of Business by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

A Lagrangian Relaxation for Golomb Rulers

Marla R. Slusky[1] and Willem-Jan van Hoeve[2]

[1] Department of Mathematical Sciences, Carnegie Mellon University
mslusky@andrew.cmu.edu
[2] Tepper School of Business, Carnegie Mellon University
vanhoeve@andrew.cmu.edu

Abstract. The Golomb Ruler Problem asks to position n integer marks on a ruler such that all pairwise distances between the marks are distinct and the ruler has minimum total length. It is a very challenging combinatorial problem, and provably optimal rulers are only known for n up to 26. Lower bounds can be obtained using Linear Programming formulations, but these are computationally expensive for large n. In this paper, we propose a new method for finding lower bounds based on a Lagrangian relaxation. We present a combinatorial algorithm that finds good bounds quickly without the use of a Linear Programming solver. This allows us to embed our algorithm into a constraint programming search procedure. We compare our relaxation with other lower bounds from the literature, both formally and experimentally. We also show that our relaxation can reduce the constraint programming search tree considerably.

1 Introduction

For some positive integer n, let $x_1, \ldots, x_n$ represent the integer positions of n marks on a ruler. We can assume that $x_i < x_j$ for all $1 \leq i < j \leq n$ and that $x_1 = 0$. A *Golomb ruler* has pairwise distinct distances between the marks, i.e., $x_j - x_i$ for all $1 \leq i < j \leq n$ are distinct. Given n, the Golomb ruler problem asks to find a Golomb ruler with minimum length x_n .

Practical applications of the Golomb ruler problem include radio communications, X-ray crystallography, coding theory, and radio astronomy [1, 2, 3, 4]. The problem continues to be very difficult to solve in practice, although it is still unknown whether it is NP-hard. Optimal Golomb rulers are only known up to $n = 26$. The optimality of rulers of 24, 25 and 26 marks was proven by a massively parallel search coordinated by `distributed.net/ogr`. The 27-mark search started in March 2009, and as of October 2012, only 65% of this project is complete.[3]

The Golomb ruler problem is a popular benchmark for discrete optimization, and for constraint programming methods in particular (it is problem prob006 in CSPLib). Several exact methods based on constraint programming have been

[3] See `http://stats.distributed.net/projects.php?project_id=27`

proposed in the literature, (e.g., [5, 6]). Other solution methods include algebraic methods (affine and projective plane constructions, [7, 8]), evolutionary algorithms [9], and hybrid methods combining constraint programming and local search [10, 11].

A crucial component of exact solution methods is producing lower bounds, which appears to be more challenging than providing upper bounds (feasible rulers). Lower bounds can help to dramatically prune an exhaustive search, but only if they can be found quickly enough. Lower bounds based on linear programming formulations were proposed in [12, 13, 14]. These three formulations were proved equivalent in [15]. Another bound was discussed in [6] and applied within a constraint programming approach for solving the problem. This bound is weaker than the LP bound, but it can be computed more quickly.

In this paper, we propose a new method for producing lower bounds, based on a Lagrangian relaxation of the problem. We show that our relaxation generalizes the bounds proposed in [6], and can produce a bound that is equivalent to the LP bound. Furthermore, we present an algorithm that allows solving the relaxation in $O(n^2 \log n)$ time for fixed Lagrangian multipliers. This allows us to efficiently approximate the LP bound using a subgradient optimization method, and apply our bound within a constraint programming search procedure. We experimentally demonstrate that in practice our method can produce bounds almost as strong as the LP bound much faster than existing methods. Moreover, we demonstrate that it can decrease the search tree size up to 91%, which translates into a solving time reduction of up to 78%.

We note that Lagrangian relaxations have been applied before in the context of CP, see for example [16, 17, 18, 19, 20, 21, 22, 23]. Our results further strengthen the idea that Lagrangian relaxations are a particularly useful method from operations research for enhancing the inference process of constraint programming. In particular, Lagrangian relaxations can help improve the representation of integrating arithmetic constraints into the *alldifferent* constraint, which is a challenging issue in constraint programming [24].

The rest of the paper is organized as follows. In Section 2 we present formal models of the Golomb ruler problem. In Section 3 we present the Lagrangian formulation, our efficient algorithm to solve the relaxation, and the subgradient optimization method. Section 4 discusses exact methods to solve the Lagrangian relaxation and relates our formulation to the formulations in [13], [15], and [6]. Section 5 contains the computational results comparing our new formulation to the formulations in [13] and [14], the current state of the art. In Section 6 we present our search algorithm and demonstrate the benefit provided by the Lagrangian relaxation bound.

2 Exact Models for the Golomb Ruler Problem

We first present a formal model of the Golomb ruler problem. In the following, we will assume that the marks take their position from a range $\{1, \ldots, L\}$ for some appropriate upper bound L.

Rather than taking the marks $x_1, \ldots, x_n$ to be our variables, we will take the $\binom{n}{2}$-many *segment lengths* $d_{ij} := x_j - x_i$ to be our variables. Then the Golomb ruler problem can be expressed as the following constraint programming (CP) model:

$$\begin{aligned} \min \quad & \sum_{k=1}^{n-1} d_{k,k+1} \\ \text{s.t.} \quad & \textit{alldifferent}(d_{12}, d_{13}, \ldots, d_{n-1,n}) \\ & d_{ij} = \sum_{k=i}^{j-1} d_{k,k+1} \qquad \text{for all } 2 \leq i+1 < j \leq n. \end{aligned} \tag{1}$$

We can alternatively express this CP model as an integer programming (IP) model, by representing the *alldifferent* constraint explicitly as a bipartite matching problem. That is, we introduce a vertex set corresponding to the pairs of marks $\{(i,j) \mid 1 \leq i < j \leq n\}$, a vertex set corresponding to the possible lengths $\{1, 2, \ldots, L\}$, and we define the complete bipartite graph between these two vertex sets. Clearly, a maximum matching in this graph corresponds to a solution to *alldifferent* [25]. For our IP model, we introduce a binary 'edge' variable such that $e_{ijv} = 1$ when the pair (i,j) induces a distance $v \in \{1, \ldots, L\}$ and $e_{ijv} = 0$ otherwise. The model thus becomes:

$$\begin{aligned} \min \quad & \sum_{k=1}^{n-1} d_{k,k+1} \\ \text{s.t.} \quad & \sum_{v=1}^{L} e_{ijv} = 1 && \text{for all } 1 \leq i < j \leq n, \\ & \sum_{i<j} e_{ijv} \leq 1 && \text{for all } v = 1, \ldots, L, \\ & \sum_{v=1}^{L} v \cdot e_{ijv} = d_{ij} && \text{for all } 1 \leq i < j \leq n, \\ & \sum_{k=i}^{j-1} d_{k,k+1} = d_{ij} && \text{for all } 2 \leq i+1 < j \leq n, \\ & e_{ijv} \in \{0,1\} && \text{for all } 1 \leq i < j \leq n,\ v = 1, \ldots, L. \end{aligned} \tag{2}$$

In this model, the first two constraints represent the bipartite matching. The third constraint establishes the relationship between the variables e_{ijv} and d_{ij}. The fourth is the requirement that each larger segment is made up of the smaller segments it contains. We note that model (2) corresponds to the formulation suggested in [12]. We will refer to it as the *matching* formulation and to its objective value as z_{matching}. We will derive our Lagrangian relaxation from this model.

3 Lagrangian Relaxation

In this section, we first present the Lagrangian formulation, which provides a relaxation for any fixed set of Lagrangian multipliers. We then show that each such relaxation can be solved efficiently. In order to find the best relaxation (corresponding to the LP lower bound), we lastly present a subgradient optimization method that approximates the optimal Lagrangian multipliers.

3.1 Formulation

We create a Lagrangian relaxation from model (2) as follows. For every pair of non-consecutive marks, that is, for all i, j such that $2 \leq i+1 < j \leq n$, we choose a coefficient $\lambda_{ij} \in \mathbb{R}$ and consider the LP resulting from moving the last constraint of the matching formulation to the objective function:

$$\begin{aligned}
\min \quad & \sum_{k=1}^{n-1} d_{k,k+1} + \sum_{i+1<j} \lambda_{ij} \left(d_{ij} - \sum_{k=i}^{j-1} d_{k,k+1} \right) \\
\text{s.t.} \quad & \sum_{v=1}^{L} e_{ijv} = 1 && \text{for all } 1 \leq i < j \leq n, \\
& \sum_{i<j} e_{ijv} \leq 1 && \text{for all } v = 1, \ldots, L, \\
& d_{ij} = \sum_{v=1}^{L} v \cdot e_{ijv} && \text{for all } 1 \leq i < j \leq n, \\
& e_{ijv} \geq 0 && \text{for all } 1 \leq i < j \leq n,\ v = 1, \ldots, L.
\end{aligned} \tag{3}$$

In this formulation we do not enforce $\sum_{k=i}^{j-1} d_{k,k+1} = d_{ij}$, but we do incur a penalty, weighted by λ_{ij}, if we do not satisfy that constraint. Note that the optimal solution for the matching formulation is still feasible in this relaxation, and gives the same objective value. Therefore, the optimal value here is *at most* z_{matching}.

We can simplify our model further by rearranging the objective function to become

$$\sum_{i+1<j} \lambda_{ij} d_{ij} + \sum_{k=1}^{n-1} d_{k,k+1} \left(1 - \sum_{\substack{i \leq k < j \\ i+1 \neq j}} \lambda_{ij} \right). \tag{4}$$

Also, recall that we did not choose $\lambda_{k,k+1}$ for any k earlier, so let us take

$$\lambda_{k,k+1} := 1 - \sum_{\substack{i \leq k < j \\ i+1 \neq j}} \lambda_{ij} \ . \tag{5}$$

Then for any fixed (λ_{ij}) satisfying equation (5), we have the simpler LP:

$$\begin{aligned} \min \quad & \sum_{i<j} \lambda_{ij} d_{ij} \\ \text{s.t.} \quad & \sum_{v=1}^{L} e_{ijv} = 1 && \text{for all } 1 \le i < j \le n, \\ & \sum_{i<j} e_{ijv} \le 1 && \text{for all } v = 1, \ldots, L, \\ & d_{ij} = \sum_{v=1}^{L} v \cdot e_{ijv} && \text{for all } 1 \le i < j \le n, \end{aligned} \tag{6}$$

This is the LP we will refer to as the Lagrangian relaxation, and we will refer to its objective value as z_{LR}. Note that the d_{ij} variables are simply an intermediate calculation. If we replace the d_{ij} in the objective function with $\sum_{v=1}^{L} v \cdot e_{ijv}$, then we can eliminate the third constraint, and so this LP represents a matching problem. This ensures that this LP has an integer solution.

Proposition 1. *For any fixed (λ_{ij}) we have $z_{\text{LR}} \le z_{\text{matching}}$, and there exists (λ_{ij}) for which $z_{\text{LR}} = z_{\text{matching}}$.*

Proof. The proposition follows from choosing (λ_{ij}) to be the dual variables of the last equation in (2). (see, e.g., [26]). □

3.2 A Combinatorial Algorithm for Solving the Relaxation

Proposition 2. *For any fixed (λ_{ij}), the Lagrangian relaxation can be solved in $O(n^2 \log n)$ time.*

Proof. What the Lagrangian relaxation LP actually represents is a matching problem where we are matching each λ_{ij} with a number in $\{1, \ldots, L\}$, and we are trying to minimize the sum of the product of the pairs. It is clear that if $\lambda_{ij} \ge 0$ for all $i < j$, then to minimize the objective value we must match the largest λ_{ij} with 1, the next largest λ_{ij} with 2, etc. (If we have some $\lambda_{ij} < 0$ then we will just take d_{ij} as large as possible making our objective value $-\infty$.) Thus our method for solving the Lagrangian relaxation will be as follows.

1 Sort (λ_{ij}) into decreasing order.

2 Let d_{ij} be the location of λ_{ij} in the sorted list.

Since our algorithm for solving the Lagrangian relaxation reduces to sorting $\binom{n}{2}$ elements, we can solve it in $O(n^2 \log n)$ time. □

3.3 Subgradient Optimization Method

In order to find (close to) optimal values for (λ_{ij}), we designed an iterative local search scheme similar to subgradient optimization methods as applied in, e.g., [27, 28]. To approximate good values for (λ_{ij}), recall that λ_{ij} is a penalty for not satisfying the constraint $\sum_{k=i}^{j-1} d_{k,k+1} = d_{ij}$. Therefore, if we solve the Lagrangian relaxation and do not satisfy the constraint for pair (i, j), we should increase the penalty λ_{ij}. Our algorithm is as follows:

1 Choose initial *stepsize*
2 Choose initial values for λ_{ij} with $i + 1 < j$ (for example, all 0)
3 Set $\lambda_{k,k+1} := 1 - \sum_{i<k<j} \lambda_{ij}$ for all $k \in \{1, \ldots, n-1\}$
4 Repeat until some stopping criterion {
5 Solve the Lagrangian relaxation
6 For each $i < j$ do
7 $$\lambda_{ij} \quad := \lambda_{ij} + \left(d_{ij} - \sum_{k=i}^{j-1} d_{k,k+1} \right) \frac{stepsize}{n^2}$$
8 Adjust *stepsize* if necessary
9 }

The performance of this algorithm highly depends on the choice and adjustment of the stepsize parameter. In our implementation, we start with a stepsize of 1 (in line 1). When an iteration results in negative values for some λ_{ij}, we divide the stepsize in half to refine the search. Otherwise, after each 5 iterations of decreasing values for z_{LR}, we multiply the stepsize by 0.999 (line 8).

Unfortunately, this algorithm does not have a natural stopping condition based on optimality of the solution. In fact, even if we use the optimal (λ_{ij}) as initial data, one iteration will return different values. Nevertheless, this algorithm produces very good approximations of z_{matching} very quickly, as we will see in Section 5.

4 Relationship with Other Formulations

In this section we investigate the relationship of our Lagrangian relaxation with other, existing, formulations for obtaining lower bounds. Throughout this section we will use λ to mean $(\lambda_{ij}) \in \mathbb{R}^{\binom{n}{2}}$; $S_{\binom{n}{2}}$ to be the set of all permutations of the numbers $\{1, 2, \ldots, \binom{n}{2}\}$ indexed by pairs (i, j) with $i < j$; $\sigma = (\sigma_{ij}) \in S_{\binom{n}{2}}$; and $\lambda \cdot \sigma = \sum_{1 \le i < j \le n} \lambda_{ij} \sigma_{ij}$.

4.1 Permutation Formulation

Our goal in the last section was

$$\min_{\sigma} \lambda \cdot \sigma$$

for a fixed λ, because this gives us a lower bound for the length of a Golomb ruler. However, our overall goal is to strengthen this bound, that is

$$\max_{\lambda} \min_{\sigma} \lambda \cdot \sigma$$

or, expressed as an LP,

$$\begin{aligned} \max \quad & z \\ \text{s.t.} \quad & \sum_{i \le k < j} \lambda_{ij} = 1 \qquad \text{for all } k = 1, \ldots, n-1, \\ & z \le \sum_{i<j} \lambda_{ij} \cdot \sigma_{ij} \qquad \text{for all } \sigma \in S_{\binom{n}{2}}. \end{aligned} \tag{7}$$

We will refer to this model as the *permutation* formulation. This formulation was also given in [13] and [15].

The correspondence between model (7) and our Lagrangian relaxation is that by solving model (7) we obtain optimal values for λ with respect to the Lagrangian relaxation, and both models will provide the same objective value. Unfortunately, solving the permutation model directly is non-trivial; we have about $\binom{n}{2}!$ constraints. However, we can apply Proposition 2 to solve it more quickly. We will iterate solving model (7) for a subset of constraints $C \subset S_{\binom{n}{2}}$:

$$\begin{aligned} \max \quad & z \\ \text{s.t.} \quad & \sum_{i \le k < j} \lambda_{ij} = 1 \qquad \text{for all } k = 1, \ldots, m-1, \\ & z \le \sum_{i<j} \lambda_{ij} \cdot \sigma_{ij} \qquad \text{for all } \sigma \in C. \end{aligned} \tag{8}$$

Our algorithm is as follows:

1 Choose any initial C
2 Solve (8) and let z be the objective value
3 Sort λ into decreasing order
4 For $i < j$ let σ_{ij} = (the position of λ_{ij} in sorted order)
5 If $(z = \lambda \cdot \sigma)$
6 Then terminate
7 Else {
8 $\quad C := C \cup \{\sigma\}$
9 $\quad$ Goto 2
10 }

The sorting algorithm and the restricted permutation model provide lower and upper bounds, respectively; optimality is proved when these bounds meet (line 5). This can serve as a systematic alternative approach to our local search.

4.2 Equation Sums Bound

We next study the relationship of the Lagrangian relaxation with the lower bounds proposed in [6]. For this, we consider the constraint $\sum_{i \leq k < j} \lambda_{ij} = 1$ in models (7) and (8). We assign a coefficient λ_{ij} to each segment of the ruler, but why should we have them summing to 1 in this way? Before we answer that question, we recall the lower bounds given in [6] by illustration with an example.

Let $n = 5$, for which the length of the ruler is given by d_{15}. If we want to bound d_{15}, we can first divide this segment into sub-segments in different ways:

$$\begin{aligned} d_{15} &= d_{12} + d_{23} + d_{34} + d_{45} \\ d_{15} &= d_{13} + d_{35} \\ d_{15} &= d_{12} + d_{24} + d_{45} \end{aligned}$$

Multiplying each equation by $\frac{1}{3}$ and adding them together gives

$$d_{15} = \frac{2}{3}(d_{12} + d_{45}) + \frac{1}{3}(d_{23} + d_{34} + d_{13} + d_{24} + d_{35}) \ .$$

Since all these numbers will be distinct naturals, we get

$$\begin{aligned} d_{15} &\geq \frac{2}{3}(1+2) + \frac{1}{3}(3+4+5+6+7) \\ d_{15} &\geq 10.333 \end{aligned}$$

There are, of course, many ways we can write out d_{1n} as a sum of smaller segments, and [6] proposes some heuristics. We will refer to bounds of this form as *equation sums* bounds. Another option we have is to weight the equations differently. For example, we could have given the first two equations weights of 0.4 and the last equation a weight of 0.2 instead of giving them all a weight of $\frac{1}{3}$. This would result in the equation

$$d_{15} = 0.6(d_{12} + d_{45}) + 0.4(d_{23} + d_{34} + d_{13} + d_{35}) + 0.2(d_{24})$$

and the corresponding bound

$$\begin{aligned} d_{15} &\geq 0.6(1+2) + 0.4(3+4+5+6) + 0.2(7) \\ d_{15} &\geq 10.4 \end{aligned}$$

We will refer to bounds of this form as *generalized equation sums* bounds.

Proposition 3. *The generalized equation sums bounds are equivalent to* z_{LR} *for an appropriate choice of* λ*.*

Proof. The weights of the equations in the generalized equation sums bound must always be distributed so that they sum to 1. This way d_{1n} always gets a coefficient of 1, and we always end up with a bound of the form $d_{1n} \geq \sum \mu_{ij} d_{ij}$ for some coefficients μ_{ij}. Note that in each equation for each $k = 1, \ldots, n-1$, there is some term that encapsulates the segment $(k, k+1)$. That is, there is

some d_{ij} such that $i \leq k < j$. Since the weights on each equation sum to one, the coefficients that encapsulate the pair $(k, k+1)$ should sum to 1. That is: $\sum_{i \leq k < j} \mu_{ij} = 1$. Then to find the minimum value of $\sum_{i<j} d_{ij}\mu_{ij}$ we simply sort (μ) into decreasing order and assign each d_{ij} the corresponding value. This is precisely what we did in Proposition 2. □

This shows that although the bound from [6] is weaker than the LP bound, it can be generalized to be as strong as the LP bound, and it gives a nice intuition for our constraint on λ.

5 Computational Results for Approximating the LP Bound

The purpose of our experimental results is twofold. First, we would like to gain insight in the performance of our approximate local search scheme relative to the systematic iterative scheme based on the permutation formulation for solving the Lagrangian relaxation. Second, we wish to evaluate our Lagrangian relaxation with the state of the art for solving the LP relaxation.

5.1 Subset Formulation

The current fastest method for solving the LP relaxation for the Golomb ruler problem was proposed by [15]. It is based on the following formulation of the lower bound, proposed in [14]. Let $\mathcal{S} = \{(i,j) : i < j\}$.

$$\begin{aligned} \min \quad & d_{1n} \\ \text{s.t.} \quad & \sum_{k=i}^{j-1} d_{k,k+1} = d_{ij} && \text{for all } 1 \leq i < j \leq n, \\ & \sum_{(i,j)\in R} d_{ij} \geq \frac{1}{2}|R| \cdot (|R|+1) && \text{for all } R \in \mathcal{P}(\mathcal{S}). \end{aligned} \tag{9}$$

We will call this the *subset* formulation. Again, this LP is too big to solve as stated since it has $O(2^{\binom{n}{2}})$ constraints. However, [15] proposes an iterative solving method in which we only include the second constraint above for some subset $\mathcal{T} \subset \mathcal{P}(\mathcal{S})$.

1 Let $\mathcal{T} = \{\{i\} : 1 \leq i \leq n\} \cup \{\{1, \ldots, n\}\}$
2 Solve (9)
3 Sort (d_{ij})
4 For $1 \leq k \leq \binom{n}{2}\{$
5 Let $T = \{(i,j) : d_{ij}$ is in within the first k positions$\}$
6 If $(\sum_{(i,j)\in T} d_{ij} < \binom{k}{2})$ then
7 $\mathcal{T} := \mathcal{T} \cup \{T\}$
8 }

This approach is the currently best known algorithm for finding the LP bound, and we will compare our proposed algorithm to this.

5.2 Implementation and Results

We implemented the Lagrangian relaxation and the subgradient method in C++, following the description in Section 3.3. It was run using C++ on an Intel core i3 processor (2.13 GHz). The times reported are the number of seconds elapsed between when the program started running and when that lower bound was found.

We implemented both the subset formulation and the permutation formulation in AIMMS. The AIMMS implementations were run on the same Intel core i3 processor. The times reported are the sums of the solve times for each call to CPLEX, i.e., we eliminate the overhead that AIMMS may add).

We ran the cases $n = 30$, 40, 50, and 60 to completion, and $n = 130$ for 600 seconds. In each case we can see from the figures that although the Lagrangian relaxation does not achieve the LP bound, it gets close to it before the subset formulation does. We also show, for reference, the constant functions $y = \text{UB}$, where UB is the best known upper bound (length of the shortest known ruler[4]) and $y = \text{LB}$ where LB is the value of the LP bound (the final value of z in all our formulations).

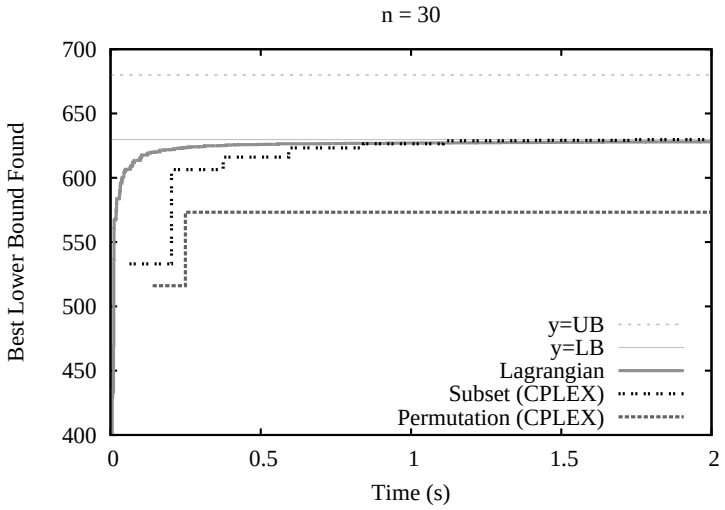

Fig. 1. Speed comparison between the permutation, subset, and Lagrangian formulations. How quickly can each find the lower bound when $n = 30$?

[4] See `http://www.research.ibm.com/people/s/shearer/grtab.html` for the list of shortest known rulers.

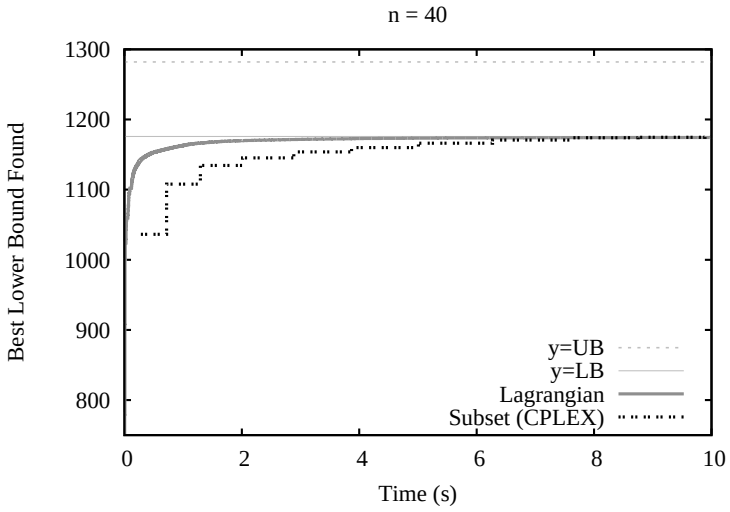

Fig. 2. Speed comparison between the subset and Lagrangian formulations. How quickly can each find the lower bound when $n = 40$?

6 CP Implementation

We implemented a CP search program that, given n and L, finds all n-mark Golomb rulers of length L. It is implemented as a set constraint problem [29] concerning two set variables: X, the set of marks in the ruler, and D, the set of distances measured by the ruler. We apply the standard subset + cardinality domain ordering, whereby we maintain a lower bound of mandatory elements (denoted by X^- and D^-) and an upper bound of possible elements (denoted by X^+ and D^+). Our constraints are as follows.

$$\begin{array}{l} X = \{x_1, x_2, \ldots, x_n\} \in [\{0, L\}, \{0, \ldots, L\}] \\ |X| = n \\ D \in [\{L\}, \{1, \ldots, L\}] \\ |D| = \binom{n}{2} \\ d \in D \iff \exists x_i, x_j \in X \text{ s.t. } x_j - x_i = d \\ x_2 - x_1 < x_n - x_{n-1} \end{array} \tag{10}$$

Our branching procedure is described in Figure 6. Line 18 ensures that between any ruler and its mirror image only one is found by this program, reflecting the last constraint in model (10).

The search strategy considers each distance $d \in \{1, \ldots, L\}$ in decreasing order and decides where and if d will be measured in the ruler.

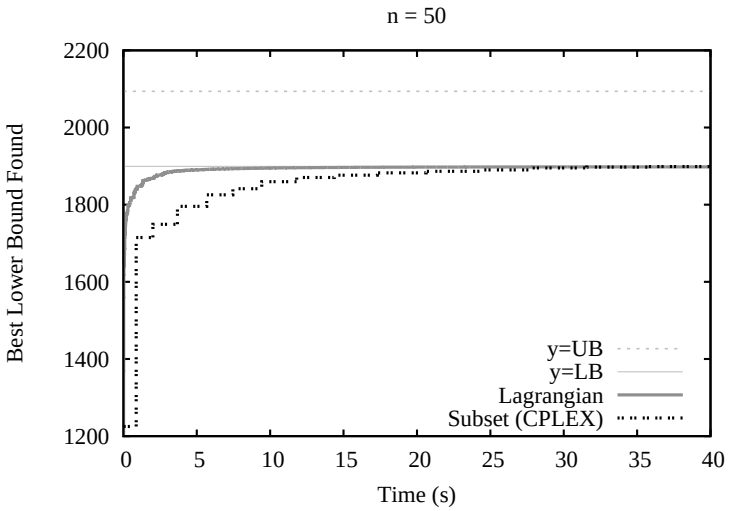

Fig. 3. Speed comparison between the subset and Lagrangian formulations. How quickly can each find the lower bound when $n = 50$?

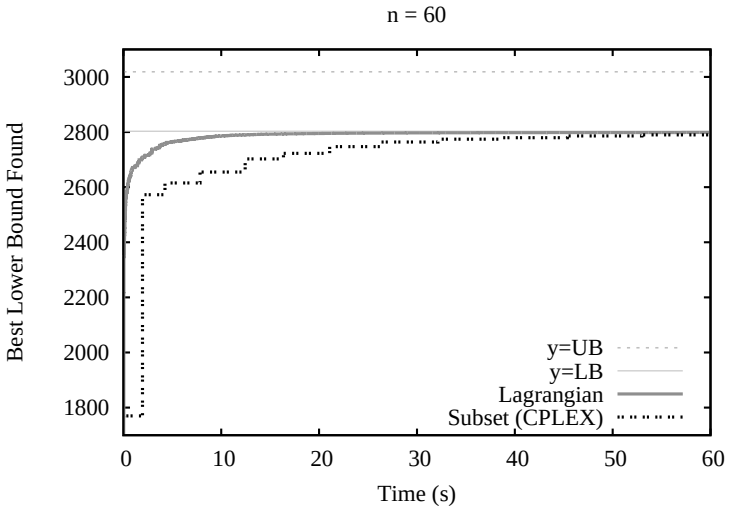

Fig. 4. Speed comparison between the subset and Lagrangian formulations. How quickly can each find the lower bound when $n = 60$?

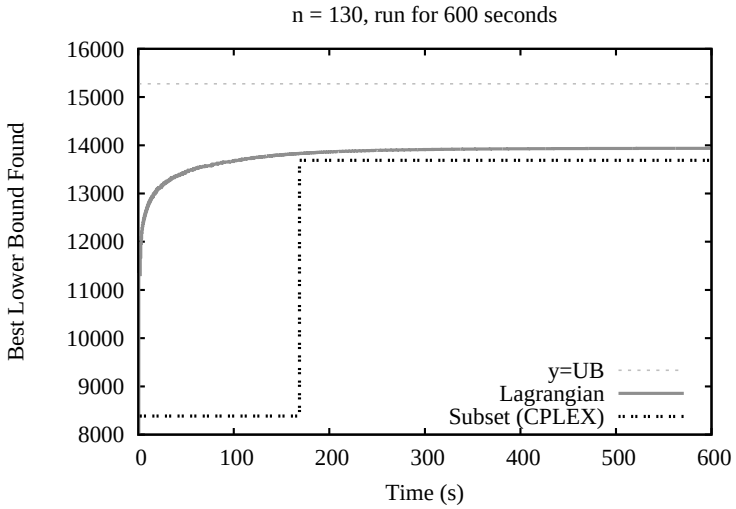

Fig. 5. Speed comparison between the subset and Lagrangian formulations. How quickly can each find the lower bound when $n = 130$?

```
1   define branch(X⁻, X⁺, D⁻, D⁺){
2     if (|X⁻| = n)
3       return X⁻
4     if (|D⁺| < (n 2))
5       return 0
6
7     D⁻ := {x_j − x_i : x_i, x_j ∈ X⁻}
8     for (x ∈ X⁻ and d ∈ D⁻){
9       if(x − d ∈ X⁺ \ X⁻)
10        X⁺ := X⁺ \ {x − d}
11      if(x + d ∈ X⁺ \ X⁻)
12        X⁺ := X⁺ \ {x + d}
13    }
14    for (x, y ∈ X⁻ with x ≡ y mod 2)
15      X⁺ := X⁺ \ {(x+y)/2}
16
17    d⁺ := max(D⁺ \ D⁻)
18    if (|X⁻| > 2)
19      if (d⁺ ∈ X⁺ \ X⁻)
20        branch(X⁻ ∪ {d⁺}, X⁺, D⁻, D⁺)
21        X⁺ := X⁺ \ {d⁺}
22    if (L − d⁺ ∈ X⁺ \ X⁻)
23      branch(X⁻ ∪ {L − d⁺}, X⁺, D⁻, D⁺)
24      X⁺ := X⁺ \ {L − d⁺}
25    branch(X⁻, X⁺, D⁻, D⁺ \ {d⁺})
26  }
```

Fig. 6. Our branching algorithm for finding Golomb Rulers

			baseline		LR1		LR2	
n	L	C/F	nodes	time	nodes	time	nodes	time
10	54	C	60,554	0.51	10,377	0.15	4,984	0.11
10	55	F	4,492	0.04	4,179	0.07	3,512	0.07
11	71	C	2,993,876	27.09	2,402,590	28.45	2,055,429	37.29
11	72	F	5,581	0.05	5,412	0.08	5,343	0.11
12	84	C	10,298,716	103.62	4,143,356	57.40	2,773,734	59.04
12	85	F	7,103,301	70.84	5,618,338	76.41	4,698,798	96.17
13	105	C	445,341,835	4782	323,717,500	5533	273,340,407	6618
13	106	F	205,714,305	2187	191,016,739	3309	177,429,879	4278

Table 1. The performance of the CP search (baseline), the Lagrangian relaxation applied at a third of the search nodes (LR1), and the Lagrangian relaxation applied at each search node (LR2). C or F denotes whether we are reporting the time/nodes to program Completion or the time/nodes to Find a ruler. We report the total number of search nodes and the solving time in seconds.

Proposition 4. *If we have already decided if and where to measure the lengths* $\{d+1, \ldots, L\}$*, and we have not decided if and where to place* d*, then the only place* d *can be measured is from 0 to* d *or from* $L-d$ *to* L*.*

Proof. Without loss of generality, suppose there is $x, x+d \in X^+$ with $0 < x < L-d$. Then since we have decided if and where to place the distance $x+d$, and we know 0 will be a mark in our ruler, we already know whether we are including the mark $x+d$. Similarly, since $d < L-x$, we already know if and where we are including the distance $L-x$ and since we are including the mark L, we also know whether we are including the mark x. If we had decided to include both x and $x+d$, then we would not need to decide on the distance d. Thus if d is the largest distance we have not decided whether or not to include, we only need to consider three possibilities: the mark d is in the ruler, the mark $L-d$ is in the ruler, or the distance d is not measured by the ruler. □

We ran three programs to test our algorithm, and the results are in Table 1. The baseline program just calls the procedure above. The other two programs start by running 2000 iterations of our subgradient optimization procedure, thus fixing our values for λ, and then call a modified version of the branch procedure, which, at line 7, uses proposition 2 to check if we have violated the LP bound. LR1 performs this check when $|X^-| \equiv 1 \bmod 3$, and LR2 performs this check at every node.

Our algorithm always reduces the size of the search tree, sometimes by as much as 91% as in Table 2. The Lagrangian relaxation does not appear to speed up the algorithm when we are searching for a ruler, but it can speed up the algorithm when we are trying to prove a ruler does not exist. Interestingly, it appears the strength of this method is correlated with the strength of the LP bound.

				LR1		LR2	
n	L	C/F	$\frac{LB}{UB}$	nodes	time	nodes	time
10	54	C	98%	82%	70%	91%	78%
10	55	F	98%	6%	-75%	21%	-75%
11	71	C	93%	19%	-5%	31%	-37%
11	72	F	93%	3%	-60%	4%	-120%
12	84	C	96%	59%	44%	73%	43%
12	85	F	96%	20%	-7%	33%	-35%
13	105	C	92%	27%	-15%	38%	-38%
13	106	F	92%	7%	-51%	13%	-95%

Table 2. Percent improvement of the Lagrangian relaxation applied at a third of the search nodes (LR1) and at each search node (LR2) over the CP search (baseline). C or F denotes whether we are reporting the time/nodes to program Completion or the time/nodes to Find a ruler. We also provide, for reference, the strength of the LP bound as $\frac{LB}{UB} = \frac{\text{LP bound}}{\text{Optimal Ruler Length}}$.

7 Conclusion

We have presented a new way to approximate the LP bound for Golomb Rulers. We have demonstrated its relationship to existing methods, and shown that we can compute the LP bound much faster using combinatorial methods.

We then used this fast computation in a search procedure, demonstrating that we can use this bound to reduce the size of the search tree and, in cases where the LP bound is strong enough, reduce the search time as well.

Bibliography

[1] Bloom, G.S., Golomb, S.W.: Applications of numbered undirected graphs. Proceedings of the IEEE **65**(4) (1977) 562–570

[2] Moffet, A.T.: Minimum-redundancy linear arrays. IEEE Transactions on Anntennas and Propagation **AP-16**(2) (1968) 172–175

[3] Gagliardi, R., Robbins, J., Taylor, H.: Acquisition sequences in PPM communications. IEEE Transactions on Information Theory **IT-33**(5) (1987) 738–744

[4] Robinson, J.P., Bernstein, A.J.: A class of binary recurrent codes with limited error propagation. IEEE Transactions on Information Theory **IT-13**(1) (1967) 106–113

[5] Smith, B., Stergiou, K., Walsh, T.: Modelling the Golomb ruler problem. In: IJCAI Workshop on Non-binary Constraints. (1999)

[6] Galinier, P., Jaumard, B., Morales, R., Pesant, G.: A constraint-based approach to the Golomb ruler problem. In: Third International Workshop on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR). (2001)

A more recent version (June 11, 2007) can be downloaded from `http://www.crt.umontreal.ca/~quosseca/pdf/41-golomb.pdf`.

[7] Singer, J.: A theorem in finite projective geometry and some applications to number theory. Transactions of the American Mathematical Society **43**(3) (1938) 377–385

[8] Drakakis, K., Gow, R., O'Carroll, L.: On some properties of costas arrays generated via finite fields. In: Information Sciences and Systems, 2006 40th Annual Conference on, IEEE (2006) 801–805

[9] Soliday, S.W., Homaifar, A., Lebby, G.L.: Genetic algorithm approach to the search for Golomb rulers. In: 6th International Conference on Genetic Algorithms (ICGA95, Morgan Kaufmann (1995) 528–535

[10] Prestwich, S.: Trading completeness for scalability: Hybrid search for cliques and rulers. In: Third International Workshop on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR). (2001)

[11] Dotú, I., Van Hentenryck, P.: A simple hybrid evolutionary algorithm for finding Golomb rulers. In: The IEEE Congress on Evolutionary Computation, IEEE (2005) 2018–2023

[12] Lorentzen, R., Nilsen, R.: Application of linear programming to the optimal difference triangle set problem. IEEE Trans. Inf. Theor. **37**(5) (2006) 1486–1488

[13] Hansen, P., Jaumard, B., Meyer, C.: On lower bounds for numbered complete graphs. Discrete Applied Mathematics **94**(13) (1999) 205 – 225

[14] Shearer, J.B.: Improved LP lower bounds for difference triangle sets. Journal of Combinatorics **6** (1999)

[15] Meyer, C., Jaumard, B.: Equivalence of some LP-based lower bounds for the Golomb ruler problem. Discrete Appl. Math. **154**(1) (2006) 120–144

[16] Sellmann, M., Fahle, T.: Constraint programming based Lagrangian relaxation for the automatic recording problem. Annals of Operations Research **118**(1–4) (2003) 17–33

[17] Cronholm, W., Ajili, F.: Strong cost-based filtering for Lagrange decomposition applied to network design. In: Proceedings of CP. Volume 3258 of Lecture Notes in Computer Science., Springer (2004) 726–730

[18] Sellmann, M.: Theoretical foundations of CP-based Lagrangian relaxation. In: Proceedings of CP. Volume 3258 of Lecture Notes in Computer Science., Springer (2004) 634–647

[19] Gellermann, T., Sellmann, M., Wright, R.: Shorter path constraints for the resource constrained shortest path problem. In: Proceedings of CPAIOR. Volume 3524 of Lecture Notes in Computer Science., Springer (2005) 201–216

[20] Khemmoudj, M.O.I., Bennaceur, H., Nagih, A.: Combining arc-consistency and dual Lagrangean relaxation for filtering CSPs. In: Proceedings of CPAIOR. Volume 3524 of Lecture Notes in Computer Science., Springer (2005) 258–272

[21] Menana, J., Demassey, S.: Sequencing and counting with the multicost-regular constraint. In: Proceedings of CPAIOR. Volume 5547 of Lecture Notes in Computer Science., Springer (2009) 178–192

[22] Cambazard, H., O'Mahony, E., O'Sullivan, B.: Hybrid methods for the multileaf collimator sequencing problem. In: Proceedings of CPAIOR. Volume 6140 of Lecture Notes in Computer Science., Springer (2010) 56–70
[23] Benchimol, P., Hoeve, W.J.v., Régin, J.C., Rousseau, L.M., Rueher, M.: Improved filtering for weighted circuit constraints. Constraints **17**(3) (2012) 205–233
[24] Régin, J.C.: Solving problems with CP: Four common pitfalls to avoid. In: Proceedings of CP. Volume 6876 of LNCS., Springer (2011) 3–11
[25] Régin, J.C.: A filtering algorithm for constraints of difference in CSPs. In: Proceedings of AAAI, AAAI Press (1994) 362–367
[26] Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley (1988)
[27] Held, M., Karp, R.M.: The travelling salesman problem and minimum spanning trees. Operations Research **18** (1970) 1138–1162
[28] Held, M., Wolfe, P., Crowder, H.: Validation of subgradient optimization. Mathematical Programming **6** (1974) 62–88
[29] Gervet, C.: Constraints over structured domains. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming. Elsevier Science Inc. (2006)