On Asymptotic Complexity of the Optimum Golomb Ruler Problem:
From Established Stochastic Methods to Self-Avoiding Walks

Franc Brglez
Computer Science, NC State University
Raleigh, NC 27695, USA
Email: brglez@ncsu.edu

Borko Bošković and Janez Brest
Faculty of Electrical Engineering and Computer Science
University of Maribor, Slovenia
Email: borko.boskovic@um.si, janez.brest@um.si

The first 8-pages are being published under IEEE Congress on Evolutionary Computation 2017
Donostia - San Sebastián, Spain, June 5-8, 2017 http://cec2017.org/
Page 3 corrects two typos. Page 9 is a tabular representation of data in Figure 1 of this paper.

Abstract—Optimum or near-optimum solutions of the Golomb ruler (gr) problem have applications in information theory, error correction, current transformers, radio frequency selection, placement of antenna arrays in astronomy, among others. In mathematics, there is a well-defined relationship of Golomb rulers and graceful graphs. A massively parallel computing project on gr has been ongoing for more than 10 years: the order-24 ruler was claimed as optimum in November 2004 (after a 4-year computational effort), followed by order-25, -26, -27 rulers in 2008, 2009, 2014. The order-28 ruler is in-progress. The distribution of waiting times, in years, such as {4, 4, 1, 5} may be impossible to predict under a variable number of processors running simultaneously.

This paper proposes a model to experimentally predict the asymptotic runtime complexity of any gr solver that returns the best-known-value (BKV) Golomb ruler defined by the paired list \((L = length, M = order)\). A subset of this list includes \{6, 4\}, \{11, 5\}, \{17, 6\}, \{25, 7\}, \{34, 8\}, \{44, 9\}, \ldots, \{680, 30\}\}. Given the number of processors \(N\) and the runtime limit \(t_{\text{lim}}\), we observe at least \(N_u \geq 100\) processors reaching the target BKV with the first-passage-time \(< t_{\text{lim}}\) and say that each such observation is uncensored. In other words, the mean runtime value we measure is based strictly on at least 100 uncensored observations from the experiment. Experiments in this paper focus on a stochastic gr-solver that implements a variation of a self-avoiding walk. In the total number of steps, the solver asymptotic walk length complexity is \(409.2 \times 10^{176}\). When measuring the number of CPU seconds on a loaded grid of 100 processors, the solver asymptotic runtime complexity is \(0.000206 \times 10^{8711}\). This solver significantly outperforms the alternative stochastic gr-solvers reported to date.

I. INTRODUCTION

The background and foundations for numerous applications of the Golomb ruler (gr) problem listed in our abstract have been comprehensively surveyed in 1977 \[1\]. Examples of specific applications can be found in \[2\].

Golomb rulers are defined with length \(L\) and order \(M\), denoted as \(gr(L, M)\). For values of \(M \leq 27\), a list of best-known-values (BKV) of \(L\) is posted in Wikipedia \[3\]. Below, we extend this table with values up to \(M \leq 30\):

\[
\text{BKV}(L, M) = \{(6, 4), (11, 5), (17, 6), (25, 7), (34, 8), (44, 9), (55, 10), (72, 11), (85, 12), (106, 13), (127, 14), (151, 15), (177, 16), (199, 17), (216, 18), (246, 19), (283, 20), (333, 21), (356, 22), (372, 23), (425, 24), (480, 25), (492, 26), (553, 27), (585, 28), (623, 29), (680, 30)\}
\]

Most remarkably, two efficient programs, implemented by Shearer in 1986 \[4\], compute a wide range of near-optimal Golomb rulers that are identical with BKV rulers in Eq. 1 except for these six rulers: \(gr(28, 7), gr(35, 8), gr(45, 9), gr(111, 13), gr(155, 15)\), and \(gr(179, 16)\). In 1979, rulers in the range of \(10 \leq M \leq 12\) were proven optimum \[5\], followed by the optimum ruler for \(M = 13\) \[6\]. In 1990, rulers in the range of \(14 \leq M \leq 16\) were proven optimum; the exhaustive computer search for \(gr(179, 16)\) took over 100 CPU hours on IBM 3090 \[7\]. Rulers in the range of \(17 \leq M \leq 19\) were proven optimum in 1998; the ruler \(gr(246, 19)\) took approximately 36200 CPU hours, normalized to a Sun Sparc Classic workstation \[8\]. The differences between the currently proven optima and the values we can readily reproduce using the two Shearer’s programs are vividly depicted in Figure 1. What stands out in subfigures 1b and 1c is the perfect overlap of BKVs that represent rulers obtained with a combination of two Shearer’s programs versus the BKVs listed in Wikipedia \[3\]. It is almost incredible that after years of massively parallel computational effort \[9\], no new and shorter values of \(L\) have been reported for rulers in the range \(20 \leq M \leq 27\). We find no scholarly publications that explain why any of the rulers with order \(M \geq 20\) are indeed optimum rather than just BKVs!

We cannot prove the optimality of a gr solution with a stochastic solver. However, we can observe and infer from its uncensored average runtime performance. While an exact solver proves that the BKV is indeed the optimum, the uncensored stochastic solver returns the same BKV — but now with the average runtime significantly less than the exact solver. The runtime gap between these two solvers increases rapidly with the increasing size of the problem. We have demonstrated this for the labs problem in \[10\] and we are demonstrating it again for the gr problem in this paper. To provide a context with runtimes listed in the previous paragraph for two rulers under exhaustive enumeration (albeit on much slower processors), we list uncensored runtime statistics for the same two rulers when running experiments on a shared grid of \(N = 100\) loaded processors (a configuration of AMD 2.1-GHz Opteron processors 6272) with the stochastic solver introduced in this paper. For \(gr(179, 16)\): runtime mean of 3023 seconds, standard deviation of 2779 seconds, minimum of 38 seconds, and maximum of 13896 seconds. For \(gr(246, 19)\): runtime mean of 16.2 hours, standard deviation of 15.3 hours.
minimum of 5.5 minutes, and maximum of 2.49 days. Since the runtime limit on the grid is \( t_{lim} = 4 \) days, the reported runtime statistics for both rulers is uncensored!

The article [11] from 1995 may well be the first attempt to use a stochastic method when searching for Golomb rulers in the range of \( 5 \leq M \leq 16 \); however, it does not succeed in finding \( \text{BKV} \) for \( M = 16 \). A number of articles on stochastic methods applied to searching for Golomb rulers over the same range of \( M \) followed. The first article that reports at least one solution with \( \text{BKV} \) for \( M = 16 \) is from 2007 [12] -- however, no runtime is reported. At least one solution with \( \text{BKV} \) for \( M = 16 \) is again reported a year later [13] -- without reporting the runtime. For a hybrid genetic algorithm published in 2010 [14] we have a report for at least one solution with \( \text{BKV} \) for \( M = 16 \) where the runtime is to be extrapolated by the reader as follows: ... around 5 hours for 11 marks, 8 hours for 12 marks, 11 hours for 13 marks, etc. The article published in 2015 [15] reports experimental results with more transparency: the success rate to reach \( \text{BKV} \) for \( M = 16 \) under the median runtime of 14.83 hours is 60%.

The paper is organized as follows: Section 2 introduces notation and definitions, Section 3 outlines the \( gr \) algorithm and the \( gr \) solver instrumentation, Section 4 summarizes asymptotic experiments and a prediction on the number of parallel processors required to reach \( \text{BKV} \) for instances with \( M \geq 20 \), Section 5 concludes with a conjecture that there may exists at least one \( gr \) which, for \( M \geq 20 \), has a \( \text{BKV} \) better than the ones listed in Eq. 1 and illustrated in Figure 1.

II. NOTATION AND DEFINITIONS
Due to page limitation, we ask the reader to also consult the related work [10] for clarifications and background about the notation and definitions used in this paper.

Fig. 1: Golomb rulers are defined with length \( L \) and order \( M \), denoted as \( gr(L, M) \). Two sources for plots of \( gr(L, M) \) are: (1) data that we reproduced within a runtime of 1 minute by combining best results from two programs created by Shearer in 1986 [4], (2) data based on the best-known-values (\( \text{BKV} \)) of \( L \), tabulated under Wikipedia [3] for the same value of \( M \). As shown, only six rulers constructed by Shearer’s programs exceed \( \text{BKV} \) posted on Wikipedia: \( gr(28, 7) \), \( gr(35, 8) \), \( gr(55, 9) \), \( gr(151, 13) \), \( gr(155, 15) \), \( gr(179, 16) \). We analyze a subset of these differences in Table I. The phase transition intervals observed in Table I raise a question whether there really are no such intervals for rulers in the range \( 20 \leq M \leq 27 \).
The $gr$ puzzle. We interprete the ruler $gr(L, M)$ as the puzzle defined in Figure 2. Each placement of $M$ pegs into $L + 1$ holes induces a complete edge-weighted graph with $M$ vertices where edge weights are defined by the graph linear arrangement. With $M$ pegs and the holes numbered from 0 to $L$, the $gr$ linear arrangement is a placement of the first peg into a fixed position 0, the last peg into a fixed position $L$, and the placement of remaining $M - 2$ pegs into any of the remaining $L - 1$ holes. Given two pegs at positions $0 < i < L$ and $0 < j < L$, the weight of each edge is simply $w_{ij} = w_{ji} = |i - j|$. As illustrated in in Figure 2, positions of up to $M - 2$ pegs are encoded with a binary coordinate of length $L - 1$. The insertion or removal of each peg at position $i$ corresponds to flipping the bit from 0 to 1 or from 1 to 0, respectively. The insertion and removal of pegs induces a sequence of dynamically changing complete graphs from 6 to 3 edges with corresponding sequences of edge weights. The most desirable sequence is not only the sequence with 0 repeated edge weights, it is the sequence where the maximum number of pegs have been inserted. The cost function we minimize during the search for the binary coordinate with BKV (or better) is defined as follows:

$$\Theta(\zeta) = \begin{cases} 
\text{numRepeats} & \text{if numRepeats} > 0 \\
- \text{coordRank} & \text{otherwise}
\end{cases} \tag{2}$$

where $\zeta$ is the binary coordinate of length $L - 1$, $\text{numRepeats}$ denotes the number of repetitions in the edge weight sequence, and $\text{coordRank}$ denotes the distance of the binary coordinate from the reference coordinate 00000000, i.e. the coordinate with rank = 0. See also Figure 3 for additional context of coordRank.

Given the formulation in Eq. 2, the cost function values $> 0$ imply that the peg arrangement has not produced a Golomb ruler while cost function values $< 0$ imply a Golomb ruler solution. Under this formulation, the coordinate $\zeta^*$ returns the global minimum of the objective function (or the solution of the puzzle) with the value of

$$\Theta(\zeta^*) = -M + 2 \tag{3}$$

The landscape graph. We maintain several views of the landscape graph, a data structure underlying the implementation of each of our stochastic combinatorial solvers; examples for a variety of problem-specific structures are also provided in [10], [16] and [17]. We use small problem instances to depict canonic views such as the example shown in Figure 3. Given the $gr$ puzzle instance in Figure 2, the principal columns in table, coordinate and weight, are generated by exhaustive enumeration of the Eq. 2. The column state is a list of symbolic names for each coordinate, values under the column rank correspond to distances of each coordinate from the reference coordinate 00000000. Symbolic names rather than specific coordinates are typically used to represent configurations or states when annotating Markov chains. The graph bidirectional edges in the landscape graph next to the table are induced by computing unit distances between the binary coordinates; we associate no self-loops with this graph. The vertex labels are state-weight pairs which can be expanded to coordinate-weight pairs $\zeta : \Theta(\zeta)$ when needed for clarity.

Formally, the landscape graph is represented with a weighted graph adjacency matrix and just like in the state transition probability graph, we can define simple walks on
Fig. 3: Canonic views of the gr puzzle instance gr(6, 4) in Figure 2. The length \( L = 6 \) induces \( 2^{6-1} = 32 \) states of complete graphs with vertices ranging from 2 to 6, each evaluated under the linear arrangement by Equation 2 which takes the 32 states encoded as a 5-bit binary coordinates, arranged in the table. The graph bidirectional edges in the landscape graph next to the table are induced by computing unit distances between the binary coordinates. The vertex labels are state-weight pairs which can be expanded to coordinate-weight pairs \( \langle \cdot, \cdot \rangle \).

The x-axis shows the vertex ranks (32 vertices, 80 edges). The y-axis shows the graph height (vertes projections from each rank). The landscape graph has vertices with minimum weight are denoted as absorbing vertices.

The landscape graph. Two of the state-weight pairs, \( Z_1 : -2 \) and \( Z_2 : -2 \), are associated with the minimum weight and are denoted as absorbing states. Since both are declared absorbing, these two vertices have incoming edges only. In the context of stochastic combinatorial optimization, the concept of absorbing states is replaced with solution coordinates, reaching the target value or BVK.

**Coordinate distance.** The distance between two binary coordinates \( a \) and \( b \) is defined as the Hamming distance:

\[
d(a, b) = \sum_{i=1}^{L-1} a_i \oplus b_i
\]

**Distance=1 neighborhood.** The distance=1 neighborhood of a coordinate \( \xi_j \) is a set of coordinates, each of length \( L - 1 \):

\[
\mathcal{N}(\xi_j) = \{ \xi'_i | d(\xi'_i, \xi_j) = 1, \quad i = 1, 2, \ldots, L - 1 \}
\]

**Contiguous walks and pivot coordinates.** Let the coordinate \( \xi_0 \) be the initial coordinate from which the walk takes the first step. Then the sequence

\[
\{ \xi_0: \xi_1: \xi_2: \cdots: \xi_j: \cdots: \xi_\omega \}
\]

is called a walk list or a walk of length \( \omega \), the coordinates \( \xi_j \) are denoted as pivot coordinates and \( \Theta(\xi_j) \) are denoted as pivot values. Given an instance of size \( L \) and its best upper bound \( \Theta^b \), we say that the walk reaches its target value (and stops) when \( \Theta(\xi_\omega) \leq \Theta^b \).

We say that the walk is contiguous if the distance between adjacent pivots is 1; i.e., given Eq. 4, we find

\[
d(\xi_j, \xi_{j-1}) = 1, \quad j = 1, 2, \ldots, \omega
\]

**Self-avoiding walks (SAWs).** We say that the walk is self-avoiding if all pivots in Eq. 6 are unique. We say that the walk is composed of two or more walk segments if the initial pivot of each walk segment has been induced by a heuristic such as random restarts. We view each walk or walk segment as a sequence of steps that chain a set of pivot coordinates:

- at each step we compute the set of all adjacent coordinates \( \mathcal{N}(\xi_j) \), also known as the local neighborhood of the pivot coordinate \( \xi_j \)
- for each adjacent coordinate \( \xi'_i \), we probe or evaluate the cost function such as Eq. 2 for its value.

There is a large family of self-avoiding walks; some are more efficient than others, depending on the problem. While for the labs problem [10], the simple wandering walk outlined below has been most efficient to date, a specialized
**meandering walk** described in Figure 4 in the next section is considerably more efficient for the \( q_r \) problem.

**Wandering Walk.** We describe the *wandering walk* in its simplest form; formal details can be found in [16] and [17].

```plaintext
1: procedure SAW.wander()
2:       while true do
3:           (1) select a random coordinate and mark it as the 'initial pivot'.
4:           (2) probe all unmarked adjacent coordinates, then select and mark the coordinate with the 'best value' as the new pivot. If multiple coordinates return the same value, a random choice is made to avoid biasing the walk.
5:           (3) continue the walk until either the 'best value' \( \leq \) 'target value' or the walk is being trapped by adjacent coordinates that are already pivots;
6:           (4) if the walk is trapped, restart the walk from a randomly selected 'new initial pivot';
7:           (5) manage the memory constraints with an efficient data structure such as a hash table.
8:       end while
9: end procedure
```

For illustrations of three ‘wandering walks’ that find all minima of the \( q_r \) problem in relatively few steps, see the landscape graph in Figure 3.

**Trapped pivots** have been observed only rarely, and only for very small instances. Issues that may arise when running this algorithm include:
- with large instances, the length of the walk may increase the number of hash collision and impact the runtime of the solver; a random restart clears the hash.
- in some domains such as the \( q_r \) problem, shorter SAW segments may be more efficient than long SAW segments – even before hash collisions become an issue.
- as the size of the problem increases, the computational cost of probing each adjacent coordinate of the pivot coordinate before making the decision on the next step of the walk can become an important factor. For any \( q_r(L, M) \), the complexity of probing the pivot and its entire neighborhood (of size \( L - 1 \)) under a simple or a naive approach is

\[
O(M^2/2 + (L - 1)M^2/2) \tag{8}
\]

In the next section we introduce an efficient approach to probing the entire neighborhood of the pivot which we call the *tableau-based probing*.

### III. SAW.MEANDER: SOLVER AND INSTRUMENTATION

We return to the landscape graph in Figure 3 to explain the *meandering walk* concept. None of the three SAWs shown in this figure are meandering walks. When specify the \( q_r \) instance as \( q_r(6, 4) \), we have \( L = 6, M = 4 \) and are searching for binary coordinates \( \Theta(\zeta^*) \) at rank \( M - 2 \) for which \( \Theta(\zeta^*) = -M + 2 \). In this case, we find by inspection two such coordinates: 01001 and 10010; both have the rank \( M - 2 = 2 \) and \( \Theta(\zeta^*) = -M + 2 = -2 \). Three examples of SAWs under the meandering walk heuristic in Figure 3 are:

- \( aF \rightarrow B \rightarrow Z1 \)
- \( E \rightarrow D \rightarrow Z2 \)
- \( C \rightarrow B \rightarrow Z1 \)

**Meandering Walk.** The meandering walk is a heuristic controlled by two coordinate parameters:

1. coordinate \( \text{rank} = M - 2 \),
2. coordinate \( \text{rank} = M - 3 \).

- initialize the first pivot as a random binary coordinate with \( \text{rank} = M - 2 \) (step 0);
- probe the pivot neighborhood strictly at \( \text{rank} = M - 3 \) only and select the 'best pivot' at \( \text{rank} = M - 3 \);
- probe the pivot neighborhood strictly at \( \text{rank} = M - 2 \) only and select the 'best pivot' at \( \text{rank} = M - 2 \);
- probe the pivot neighborhood strictly at \( \text{rank} = M - 3 \) only and select the 'best pivot' at \( \text{rank} = M - 3 \);
- continue with the meandering process between ranks \( M - 2 \) and \( M - 3 \) until BKV is found at \( \text{rank} = M - 2 \) as expressed in Eq. 3.

**Solver and Instrumentation.** The *meandering walk* implementation of the current solver (ogr1) is based on the pseudo-code of the algorithm in Figure 4. While not declared explicitly, the coordinate:value pair \( \zeta_0: \Theta(\zeta_0) \) (as the initial solution) is always based on random seed \( \sigma_0 \) and the random coordinate \( \zeta_0 \) with \( \text{rank} = M - 2 \). In other words, the initialization is defined by the meandering walk heuristic!

The algorithm is instrumented with counters such as cnt-Proof, cntTrapped, cntRestart, total number of steps and decision variable such as isCensored, targetReached. For each run under different initial seeds, results are returned in the variable Table for statistical evaluation after completion of \( N \) runs. The shaded areas in the two procedures that return a new pivot for each of the two ranks represent the two subsets of coordinate neighborhoods that are being evaluated under the tableau method briefly described next.

**On complexity of probing the Pivot Neighborhood.** The complexity of probing the pivot and its entire neighborhood under what we denote as a simple or a naive approach is expressed in Eq. 8: the cost of the neighborhood probing of size \( L - 1 \) is dominated by \( O((L - 1) \times M^2/2) \). Data structures that implement a tableau-based probing reduce the complexity of the neighborhood probing from \( O((L - 1) \times M^2/2) \) to:

- \( O((L - 1)M) \) under the wandering walk, and to
- \( O((L - M + 2)M + (M - 3)M) \) under the meandering walk.

Under the wandering walk, each pivot visits the entire neighborhood of \( L - 1 \) coordinates. Under the meandering walk, the pivot at \( \text{rank} = M - 2 \) visits the neighborhood of \( M - 3 \) coordinates and the pivot at \( \text{rank} = M - 3 \) visits the neighborhood of \( L - M + 2 \) coordinates. Additional details about the tableau-based data structures are provided in [18].

### IV. SUMMARY OF EXPERIMENTS

Our experiments follow the methodology that rigorously established the asymptotic performance model in [10]. We observe
Fig. 4: Pseudo-code of the algorithm that implements the meandering self-avoiding walk heuristic of the \(g\varepsilon(L, M)\) solver ogr1. Under this heuristic, \(s_0\) is a random initial binary coordinate based on the random seed \(\sigma_0\) such that \(rank\) of the coordinate is \(M - 2\). The shaded areas in the two procedures that return pivots for \(rank = M - 2\) and \(rank = M - 3\) respectively, highlight two coordinate neighborhood subsets evaluated under the tableau-based probing.
Fig. 5: Asymptotic experiments and predictions of observed 'hits' after 4-day runs on the grid of 100 processors.
a \texttt{gr} experiment with \( N \) processors invoking the same solver with different initial seeds and define as an \textit{uncensored sample the first-passage-time < \( t_{\text{lmt}} \)} recorded by each processor upon reaching the target \texttt{BKV}. As soon as the number of uncensored samples \( N_u \) reaches or exceeds 100, we declare the experiment completed for the target \texttt{BKV}.

The plots A-B-C-D in Figure 5 summarize asymptotic experiments performed serially on a single 2.93-GHz processor. The plots E-F summarize experiments on a grid of 100 shared processors (AMD 2.1-GHz Opteron processors 6272). Two experiments on the single processor, one with length of \texttt{walkSegmentLimit} = 8 \*(\( L - 1 \)) to induce random restarts if needed, and one with unlimited length of the walk segment (U) demonstrate that for \texttt{gr} problem, the solver configuration \texttt{ogr1_8} is preferable. The main results with the solver \texttt{ogr1_8} are these models:

- \texttt{walkLengthMeanSingle} = 58.6973 \* 1.1007\(^L\)
- \texttt{runtimeMeanSingle} = 0.000025 \* 1.1141\(^L\)
- \texttt{walkLengthMeanGrid} = 409.2 \* 1.0762\(^L\)
- \texttt{runtimeMeanGrid} = 0.0000206 \* 1.0871\(^L\)

Experiments on the grid include uncensored experiments with 4 more rulers which would take far too long to run on a single processor. While the model for \texttt{runtimeMeanGrid} represents runtime on \( N \) loaded processors, the model for \texttt{walkLengthMeanGrid} is platform-independent. Since it covers the range of 9 \( \leq M \leq 19 \), \texttt{walkLengthMeanGrid} can serve as a \textit{currently most accurate model} for platform-independent comparisons with other \texttt{gr} solvers. Details about additional experiments will be available in [18].

V. CONCLUSIONS

We return to Figure 1 and conclude with a conjecture:

\textit{There exists at least one \texttt{gr} which, for \( M \geq 20 \), has a \texttt{BKV} better than the ones listed in Eq. 1. As shown in Figure 1a there are six such \texttt{BKV}s improvements on rulers with \( M < 20 \), computed by Shearer’s programs [4].}

The pursuit of this conjecture will require:

- theoretical research to explain why any of the rulers with order \( M \geq 20 \) are optimum rather than just \texttt{BKV}s.
- experimental research for breakthrough algorithms.
- larger grids of processors as expressed in the plot \( F \) of Figure 5: with the solver \texttt{ogr1_8}, close to 1,000,000 processors are required to run for 4 years to reliably confirm \texttt{BKV} for the ruler \texttt{gr}(425, 24).

Results of two experiments in Table I show a direction that has merits of its own. Starting with \texttt{BKV} computed with Shearer’s program [4], the solver \texttt{ogr1_8} returns not only most if not all equivalent rulers but also, most importantly, the time limit \( t_{\text{lmt}} \) for the next ruler with shorter length. After traversing efficiently several phase transitions, we stop on finding the single solution only.

<table>
<thead>
<tr>
<th>Table I: Occurrences of phase transitions in the \texttt{gr} problem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{rulerID}</td>
</tr>
<tr>
<td>111,13</td>
</tr>
<tr>
<td>110,13</td>
</tr>
<tr>
<td>109,13</td>
</tr>
<tr>
<td>108,13</td>
</tr>
<tr>
<td>108,12</td>
</tr>
<tr>
<td>107,13</td>
</tr>
<tr>
<td>107,12</td>
</tr>
<tr>
<td>106,13</td>
</tr>
<tr>
<td>106,12</td>
</tr>
<tr>
<td>155,15</td>
</tr>
<tr>
<td>154,15</td>
</tr>
<tr>
<td>154,14</td>
</tr>
<tr>
<td>153,15</td>
</tr>
<tr>
<td>152,15</td>
</tr>
<tr>
<td>151,15</td>
</tr>
<tr>
<td>151,14</td>
</tr>
</tbody>
</table>

Acknowledgement. This work was supported in part by the Slovenian Research Agency under program P2-004: Computer Systems, Methodologies, and Intelligent Services.

REFERENCES

TABLE II: Golomb ruler solutions generated by Shearer’s programs conap.f and conpp.f (AP = affine plane, PP = projective plane) [4]. Solutions from each program are compared for each $M$ and the solution with shortest length $L$ is tabulated. The total runtime to compute the contents of this table is about 1 minute.

<table>
<thead>
<tr>
<th>$L_{BKV}$</th>
<th>$M$</th>
<th>$L_{method}$</th>
<th>$method$</th>
<th>isBKV</th>
<th>diff</th>
<th>diff(%)</th>
<th>ruler</th>
<th>method</th>
<th>(BKV = best-known-value of ruler length $L$, given the ruler order $M$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>AP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>AP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>AP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>1.4, 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>AP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>1, 4, 9,11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>6</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.1, 4,10,12,17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>7</td>
<td>28</td>
<td>AP</td>
<td>NO</td>
<td>3</td>
<td>12.00</td>
<td>0.6, 9,10,21,23,28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>8</td>
<td>35</td>
<td>PP</td>
<td>NO</td>
<td>1</td>
<td>2.94</td>
<td>0.7, 10,16,18,30,31,35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>9</td>
<td>45</td>
<td>PP</td>
<td>NO</td>
<td>1</td>
<td>2.27</td>
<td>0.3, 9,16,28,21,35,43,45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>10</td>
<td>55</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.1, 6,10,23,26,34,41,53,55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>11</td>
<td>72</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.1, 9,19,24,31,52,56,58,69,72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>12</td>
<td>85</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.2, 6,24,29,48,43,55,68,75,76,85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>13</td>
<td>111</td>
<td>AP</td>
<td>NO</td>
<td>5</td>
<td>4.72</td>
<td>0.6, 18,19,23,39,49,64,71,73,100,108,111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>14</td>
<td>127</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.5, 28,38,41,49,50,68,75,92,107,121,123,127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>151</td>
<td>15</td>
<td>155</td>
<td>PP</td>
<td>NO</td>
<td>4</td>
<td>2.65</td>
<td>0.7, 13,16,30,38,50,77,96,98,122,140,150,151,155</td>
<td></td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>16</td>
<td>179</td>
<td>PP</td>
<td>NO</td>
<td>2</td>
<td>1.13</td>
<td>0.9, 21,43,47,61,66,67,96,103,135,151,166,168,176,179</td>
<td></td>
<td></td>
</tr>
<tr>
<td>199</td>
<td>17</td>
<td>199</td>
<td>AP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.5, 7,17,52,56,67,80,81,100,127,138,159,165,168,191,199</td>
<td></td>
<td></td>
</tr>
<tr>
<td>216</td>
<td>18</td>
<td>216</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.2, 10,22,53,56,82,83,89,98,138,148,153,167,188,192,285,216</td>
<td></td>
<td></td>
</tr>
<tr>
<td>246</td>
<td>19</td>
<td>246</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.4, 13,15,42,56,59,77,93,116,126,138,146,174,214,221,240,245,246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>283</td>
<td>20</td>
<td>283</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.24, 30,43,55,71,75,89,104,125,127,162,167,189,206,215,272,275,282,283</td>
<td></td>
<td></td>
</tr>
<tr>
<td>333</td>
<td>21</td>
<td>333</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.4, 23,37,40,48,68,78,138,147,154,189,204,238,250,251,256,277,309,331,333</td>
<td></td>
<td></td>
</tr>
<tr>
<td>356</td>
<td>22</td>
<td>356</td>
<td>AP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.1, 9,14,43,70,106,122,124,128,159,179,204,223,253,263,270,291,330,341,353,356</td>
<td></td>
<td></td>
</tr>
<tr>
<td>372</td>
<td>23</td>
<td>372</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.6, 22,24,43,56,95,126,137,146,172,173,201,213,258,273,281,306,311,355,365,369,372</td>
<td></td>
<td></td>
</tr>
<tr>
<td>425</td>
<td>24</td>
<td>425</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.9, 33,37,43,97,122,129,140,147,152,191,205,268,252,278,286,326,332,353,368,384,403,425</td>
<td></td>
<td></td>
</tr>
<tr>
<td>480</td>
<td>25</td>
<td>480</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.12, 29,39,72,91,146,157,160,161,166,191,207,214,258,290,316,354,372,394,396,431,459,467,480</td>
<td></td>
<td></td>
</tr>
<tr>
<td>492</td>
<td>26</td>
<td>492</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.5, 17,28,36,52,62,106,136,149,174,178,234,241,243,289,292,307,329,368,382,388,409,459,491,492</td>
<td></td>
<td></td>
</tr>
<tr>
<td>553</td>
<td>27</td>
<td>553</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.3, 15,41,66,95,97,106,142,152,228,221,225,242,295,338,338,354,382,388,402,415,486,504,523,546,553</td>
<td></td>
<td></td>
</tr>
<tr>
<td>585</td>
<td>28</td>
<td>585</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.3, 15,41,66,95,97,106,142,152,228,221,225,242,295,338,338,354,382,388,402,415,486,504,523,546,553</td>
<td></td>
<td></td>
</tr>
<tr>
<td>623</td>
<td>29</td>
<td>623</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.7, 11,31,43,53,100,121,144,158,282,229,229,268,284,285,356,371,396,416,430,465,467,528,582,590,595,620,623</td>
<td></td>
<td></td>
</tr>
<tr>
<td>680</td>
<td>30</td>
<td>680</td>
<td>PP</td>
<td>YES</td>
<td>0</td>
<td>0.00</td>
<td>0.12,32,39,49,82,85,100,147,166,286,207,211,286,302,310,316,344,388,399,462,475,500,529,531,552,623,645,671,680</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>