New Algorithms for Golomb Ruler Derivation and Proof of the 19 Mark Ruler

Apostolos Dollas, William T. Rankin, David McCracken
Dept. of Electrical Engineering
Duke University
Durham, NC 27708-0291

November 9, 1995

Abstract

Two new algorithms for Golomb ruler derivation were developed and are presented together with the previously published standard algorithm. One of the new algorithms was used to prove computationally the optimality of three rulers. Two of these were previously proven but yet unpublished, and the authors' independent derivation confirmed these results. The last ruler, of 19 marks and size 246, was known to be near optimal and was computationally proven optimal in this work.

Keywords: Golomb Ruler, Algorithm, Parallel Implementation

1 Introduction

The term "Golomb Ruler" is derived from the relevant work by Professor Solomon W. Golomb of the University of Southern California [3]. These rulers are numerical sequences or graphs with application in a wide variety of fields including radio communications, x-ray crystallography, coding theory, and radio astronomy.

The search for Golomb Rulers requires combinatorial algorithms whose bounds can be shown to grow geometrically with respect to the solution size [16]. This has been a major limitation in the discovery of new rulers since each new ruler is by necessity larger that its predecessor. The search is bounded, however, and therefore the problem is solvable [11]. By carefully reviewing existing sequential algorithms and applying parallel programming techniques, the problem can be reduced to a workable solution. Toward that end, new algorithms were derived for efficient hardware and parallel software derivation of new rulers.

Using these new algorithms, the optimality for 17, 18 and 19 mark rulers was proved computationally. The first two were proven prior to the authors' independent efforts by Mr. Olin Sibert and communicated directly to Prof. Golomb. The optimality of the 19 mark ruler was proved first by the authors and reported to Prof. Golomb in February, 1994.

This paper presents new algorithms and the rulers which were computationally proven optimal using one of the new algorithms. Since most recent papers on Golomb rulers do not provide any introduction to the subject, the remainder of this section introduces Golomb rulers and their applications, whereas Section 2.1 presents the standard reference algorithm for ruler derivation. The remainder of Section 2 presents new algorithms, and Section 3 presents search space reduction techniques. Section 4 lists previously known rulers and Section 5 presents the computationally proven optimal rulers of sizes 17, 18, and 19. Finally, conclusions are presented.

$$
\begin{array}{ccccc}
d1_1 & & d1_2 & & d1_3 \\
& d2_1 & & d2_2 & \\
& & d3_1 & &
\end{array}
$$

Figure 1: A Difference Triangle

$$
\begin{array}{ccccccc}
0 & & 1 & & 4 & & 6 \\
\hline
& 1 & & 3 & & 2 & \\
& & 4 & & 5 & & \\
& & & 6 & & &
\end{array}
$$

Figure 2: A Difference Triangle Example

1.1 Definitions

It is necessary to define some basic terms used to describe the characteristics of Golomb Rulers. Following this, a formal definition of a Golomb Ruler is given.

1.1.1 Marks and Length

A Golomb Ruler consists of ordered series of integer numbers [7]. These numbers are referred to as *marks*, and correspond to positions on a linear scale. The difference between the values of any two marks is called the *distance* between those marks. The difference between the largest and smallest number is referred to as the *length* of the ruler, and corresponds to the largest distance for that ruler. The first mark of the series is by convention at position zero.

1.1.2 Difference Triangle

In evaluating Golomb Rulers, it is very helpful to establish a data structure to contain all the distances measured by that ruler. A difference triangle is a grid of numbers where each number represents the distance between a specific pair of marks. For a ruler of m marks, there will be $m(m-1)/2$ entries in the difference triangle. An example of a difference triangle is shown in Figure 1.

The terms $d1_x$, $d2_x$ and $d3_x$ are the first order, second order and third order differences respectively. These differences are defined by the equations :

$$d1_x = M_{x+1} - M_x \tag{1}$$

$$d2_x = d1_x + d1_{x+1} \tag{2}$$

$$d3_x = d2_x + d2_{x+1} - d1_{x+1} \tag{3}$$

The equations for higher order differences are simply extensions of the third order difference equation. The first order differences are the distances measured between every pair of adjacent marks in the ruler. The second order differences are the distances measured between marks placed two apart on the ruler. A ruler with m marks will have $m-1$ first order differences, $m-2$ second order difference, and so on, up to a single $m-1$ order difference.

Figure 2 shows an example of a difference triangle for a simple ruler with four marks. The first row of the figure represents the mark positions for the ruler. The difference triangle elements are located below the horizontal line.

1.1.3 Golomb Ruler

A Golomb Ruler is defined as an ordered numeric sequence whose corresponding difference triangle contains no duplicate elements. In other words, the distance between any two marks in a Golomb Ruler is unique for that ruler. It is important to note that while this definition does not place any restriction on the length of the ruler, the cases that are of interest to researchers are the rulers with minimum lengths.

1.1.4 Perfect Golomb Ruler

A Perfect Golomb Ruler is defined as a Golomb Ruler whose difference triangle contains all numbers between one and the length of the ruler. Since for an N mark ruler, there are $\binom{N}{2}$ entries in the difference triangle, the length of an N mark Perfect Golomb Ruler is $(N-1)N/2$.

1.1.5 Proof of Maximum Perfect Ruler

Prof. Golomb has presented a simple proof that no Perfect Golomb Rulers exist with more than four marks [3]. This proof is described below:

1. Given an N mark ruler, there are $\binom{N}{2}$ possible distances measured.
2. The overall length of a ruler defined as the maximum length measured by the ruler. A perfect ruler is defined as measuring all distances from one to the maximum length of the ruler. If no distances are omitted from the series then the maximum length of a perfect N mark ruler will be the $(N-1)N/2$.
3. Looking at the distances, the first $(N-1)$ distances sum up to the total allowed length of the ruler
$$\sum_{i=1}^{N-1} i = \frac{(N-1)N}{2}$$
and therefore, these distances must form the first row of the difference triangle.
4. In order to place the distance of length 1 in the first row of the difference triangle, it must be positioned so that when combined with one of the spaces adjacent to it, it does not form a second order distance less than length N, since all distances from 1 to $N-1$ are already contained in the first row.
5. Therefore the only possible space for the distance 1 is adjacent to the $N-1$ space, forming a distance of N in the second row of the difference triangle. Since the distance 1 cannot be adjacent to any other spaces, it must therefore be positioned on one end of the first row.
6. The distance of length 2 must now be placed in the first row. It cannot be placed next to a distance of less than $N-2$, since that would form a second order distance less than N. Furthermore, it cannot be placed next to an $N-2$ distance, since this would produce a second order distance of N, which already exists.
7. Therefore the only possible position for the distance of length 2 is adjacent to the $N-1$ space. Like the 1 distance, the 2 distance must also be on one end of the first row.

8. Since the 1 and 2 length distances must both be adjacent to the $N-1$ space, then they must be located on either side of that space.
9. Since the 1 and 2 length distances must also lie on the ends of the ruler, no more first order distances may exist.
10. Thus the Perfect Golomb Ruler can have at most three distances in the first row of the difference triangle, which is produced by a four mark ruler. Rulers of greater than four marks cannot be Perfect, since it will violate one of the above conditions.

1.1.6 Optimum Rulers

Since Perfect Rulers beyond four marks cannot exist, longer rulers are described in terms of whether or not they are Optimum. An Optimum Golomb Ruler is defined as being the shortest length for a given number of marks. As will be shown, there may exist multiple different Optimum Golomb Rulers for a given number of marks.

There also exist what are known as Near-Optimum Rulers which are simply Golomb Rulers whose length is within several units of the corresponding Optimum Ruler.

1.2 Applications

So far, we have established the characteristics of a Golomb Ruler, but we have not indicated the purpose of these rulers beyond that of a mathematical curiosity. In fact, the qualities of a Golomb Ruler that we have established find their application in a wide variety of fields. In this section we will review a selection of papers on the application of Golomb and Near Golomb Rulers to real world applications.

1.2.1 Radio Communications

Perhaps the earliest reference to Golomb Rulers, although not by that name, was in the 1953 article by W. Babcock regarding third and fifth order intermodulation distortion among consecutive channels in a radio band [1]. Babcock explores methods of channel placement to reduce or eliminate third and fifth order distortion. After reviewing several algebraic and random channel distribution methods, Babcock presents the concept of placing each channel within the frequency spectrum on intervals corresponding to the marks on a Golomb Ruler. Third order interference between channels is eliminated through this approach. Fifth order interference may also be substantially reduced [12].

An interesting result is the fact that Babcock presents a table for Golomb rulers up to ten marks, and all of the rulers up to length eight are optimum. The nine and ten mark rulers that he presents are near optimum.

1.2.2 X-Ray Crystallography

X-ray analysis of crystal structures can produce ambiguities when two different crystal lattice structures produce identical diffraction patterns. Golomb Ruler sequences were used to model the lattice structure and resolve these ambiguities, based upon the empirical observation that no two equal length Golomb Rulers will have the same difference set [2].

Although this observation has been disproved for a single pair of six mark rulers, no counter example for rulers of greater than six marks has been found, and this approach is still being used. In fact, Professor Golomb has offered an award of one hundred dollars to anyone who can produce a counter-example with greater than six marks [4].

6		11		13	8		17		18	3		15		19
	5		7			9		10			12		16	
		2					1					4		

Figure 3: A Difference Triangle Set

1.2.3 Self-Orthogonal Codes

Golomb Rulers have been used extensively in the field of coding theory for error detection and correction. Sets of rulers are used to generate self-orthogonal codes that play an important role in communications [11][10].

These codes use what is known as a *difference triangle set* (DTS). A DTS consists of a group of rulers all having the same number of marks. A DTS has the additional quality that the difference triangle for each of the members of the set contains values unique to the set. An example of a DTS is shown in Figure 3 for the ruler set $\{\{0,6,11,13\},\{0,8,17,18\},\{0,3,5,19\}\}$.

The individual rulers that constitute DTS's are used as parity encoding vectors, with marks of the rulers corresponding to bit positions. No two vectors will contain two bits set in the same bit positions, since to do so would mean that the vectors contained a common difference. Further discussion on DTS's can be found in [15] and [9].

1.2.4 Linear Arrays

Linear antenna and sensor arrays have played an important part in a wide variety of applications, most notably communications and radio astronomy.

Radio Astronomy makes use of Golomb Rulers in the following manner. In locating a distant radio source, it is essential to determine the angle between the incoming wavefront and the antenna array baseline. The antennas are all set to receive the same frequency, and the exact time of reception of the incoming signal can be measured and compared between all pairs of antennas. The direction of the incoming signal can be determined by analyzing the phase difference between the pairs of antennas.

Maximum accuracy is gained when no two antennas are the same distance apart, that is, when all the elements of the their difference triangle are unique. By placing the antennas at the marks of a Golomb Ruler, this condition is guaranteed. Furthermore, by using optimum or near optimum Golomb Rulers, the total length of the array element is minimized. This is an important concern when we consider the geographical distances utilized in radio astronomy, as well as the cost of the individual array elements [13][6].

Another application of Golomb Rulers to radio astronomy is in the selection of scanning frequencies. The purpose of this application is not so much to locate distant sources, as it is the location of the antennas themselves. Using a single pair of antennas, and scanning a set of frequencies from a distant point source the location of the antenna elements can be determined to an accuracy of centimeters across continental distances [3]. By assigning the scanning frequencies to correspond to the marks on a Golomb Ruler, maximum accuracy is guaranteed for a minimum number of scanning frequencies.

1.2.5 PPM Communications

The growing use of optical communications has produced an increased interest in pulse phase modulation (PPM) as a form of communications. In PPM communications, a serial input sequence is divided up into a series of equal length *frames*. Each frame is subdivided into a number of *slots*. Within each frame a single pulse may be placed within one of the slots. The position of the pulse within the frame communicates the information.

As we can see, proper frame synchronization is essential to assure error free communications. Frame synchronization is achieved through comparing the input stream with a stored sequence and generating a correlation factor based upon the summation of the compared inputs.

A desirable PPM sequence maximizes this correlation factor when the frames are synchronized and minimizes the correlation for any off-shifted sequence. Gagliardi, Robbins and Taylor [14] present the case that maximum correlation can be achieved if no spacing between any pair of incoming pulses is repeated for any other pair. This is first requirement of a Golomb Ruler. The authors go on to show how optimum and near optimum rulers maximize the correlation factor while minimizing the overall sequence length.

The correlation of serial pulse trains also finds application in radar coding [2].

1.3 Summary

As we have seen, Golomb Rulers find application in a wide field of subjects. However, while many rulers have been found and documented for these applications, the actual generation and verification of these rulers has rarely been discussed. In fact, the complexity involved in generating these sequences has prevented the verification of optimum rulers beyond eighteen marks as of September 1993, although only Optimum Golomb Ruler up to sixteen marks have been published [16]. In the following section, we will review the algorithms used to search for optimum and near optimum rulers.

2 Algorithms

This section will present some of the various algorithms used to generate and verify Optimum Golomb Rulers.

2.1 Scientific American Algorithm

This algorithm was presented in the December 1985 issue of Scientific American [3]. The basic algorithm is composed of two sections, a ruler generation procedure and a Golomb verification procedure. The generation procedure constructs a sequence of rulers each of which are passed to the verification procedure which determines whether or not the sequence meets the criteria of a Golomb Ruler. The following sections describe each routine in turn.

2.1.1 Exhaust

The generation procedure, called Exhaust, requires two parameters. The first parameter is the number of marks contained in the desired Golomb Ruler. The second parameter sets an upper bound on the length of the Golomb Ruler being searched for. If this boundary is set too low, no ruler will be found.

The Exhaust procedure is recursive in nature. A Golomb Ruler candidate is constructed by taking an existing N mark Golomb Ruler and appending a new mark onto the right side of the ruler, generating an $N+1$ mark ruler. An example of this is shown in Figure 4.

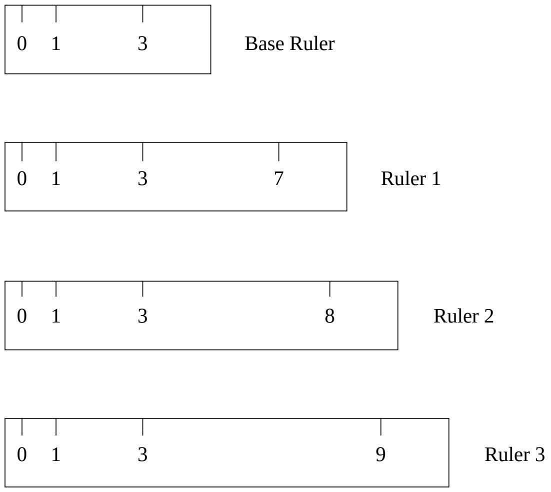

Figure 4: Example of Ruler Generation in Sci. Am. Algorithm

The Exhaust procedure does not keep track of the mark positions, but rather the spaces between adjacent marks, which are stored in an array called *SPACES*. These values represent the first row of the difference triangle for that ruler.

Exhaust begins by initializing the first element in *SPACES* to the distance 1. It then proceeds to the next distance in *SPACES* and, starting at a value of 1, increments this value until the distances measured by the first two entries are unique. It then repeats this process for the next element and so on. If at any point the total distance measured by the elements in *SPACES* exceeds the maximum ruler length, then the algorithm will back up one element and increment that element, and add new distances from there.

When the procedure places its last mark and finds a ruler of the target length, it simply prints this information out and continues the search, thus guaranteeing an exhaustive traverse of all possible ruler combinations.

A flowchart of the Exhaust algorithm is shown in Figure 5.

The Exhaust procedure is guaranteed to produce all combinations of marks within the search space determined by the number of marks and the limiting length of the ruler being searched for. The major drawback of this algorithm is that it requires a maximum value to limit the search space. If this maximum is too low, a ruler will never be found. If it is too high, then large amounts of processing will be wasted in traversing needless search space.

2.1.2 Checker

The ruler verification procedure, Checker, is the routine called by Exhaust to check if a series of marks meet the first requirement for Golomb Rulers.

The body of Checker is two nested loops that will compute every possible distance measures by the first N elements of the *SPACES* array. It uses the distance computed as an index into an array of boolean values. If the array element indexed by the distance is already set to true, then the distance being checked has already been measured by that set of marks and the sequence is not a Golomb Ruler. The procedure aborts at this point, returning a result of false. If the distance array element is clear, then the procedure sets that element to true and goes on to process the next pair of marks. If there are no conflicts after all the distances have been computed, then Checker returns a value of true.

The drawback of this routine is the fact that for every ruler, Checker must generate the

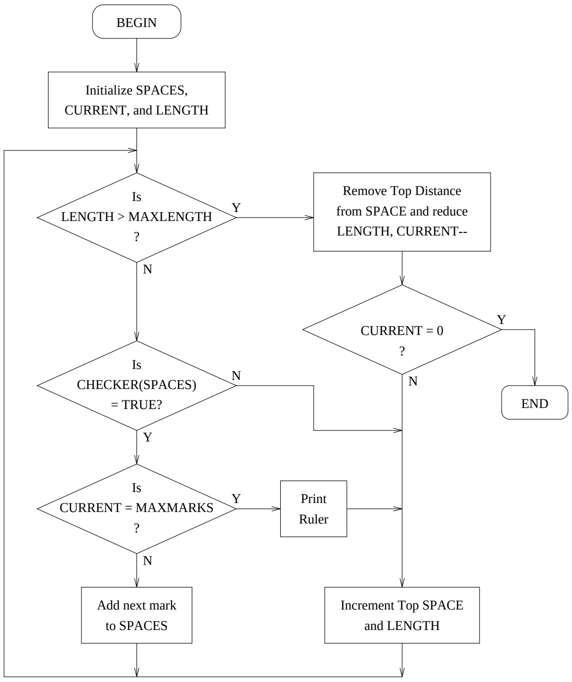

Figure 5: Flowchart of Exhaust Algorithm

complete set of distances for all the N marks. As we will see later, it is more efficient to compute only those distances added to the ruler by the new mark.

2.2 Token Passing Algorithm

This algorithm was developed by A. Dollas at Duke University. The main motivation for its development was to achieve two goals: to find the first optimum ruler without exploring a search space larger than the (unknown) ruler's size, and, to develop an algorithm which is amenable to hardware VLSI implementation with simple computational elements and a distributed control mechanism. Like the Sci. Am. Algorithm, it consists of two parts. The first section of the code generates a series of potential rulers with the second section checking to see if they are Golomb Rulers.

Unlike the previous algorithm, the token passing algorithm does not build up a ruler by appending new marks on to the end of an existing ruler. Rather, it manipulates all of the specified marks simultaneously, progressing through all possible mark positions until one is found that meets the criteria for a Golomb Ruler.

A flowchart for the for the algorithm is shown in Figure 6.

The outer loop of the main routine consists of two procedures. The Iterate procedure generates a potential Golomb Ruler. The Diffncheck procedure checks to see it the proposed ruler meets the criteria for a Golomb Ruler. If Diffncheck returns a true result then an Optimum Golomb Ruler has been found and the routine prints the results and exits. If a ruler is not found, then Iterate is called again to generate the next ruler.

The Iterate subroutine starts with a set of marks, where each mark is at its minimum position. The minimum position for a mark is the length of the Optimum Golomb Ruler that corresponds to that mark position. For example, the minimum placement for the sixth mark would be at position seventeen. The position of the TOKEN mark is the second mark since the first mark is

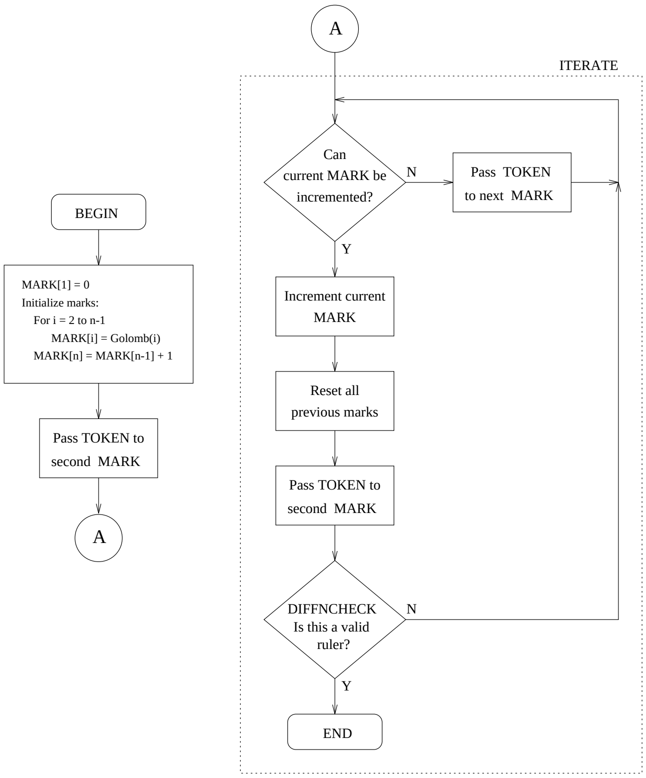

Figure 6: Flowchart for Token Passing Algorithm

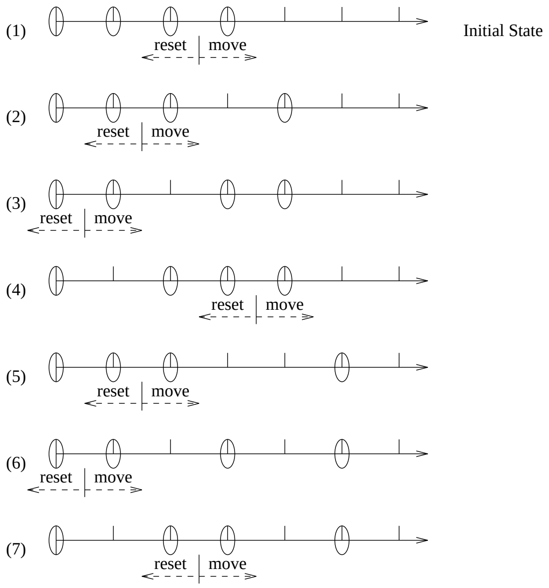

Figure 7: Example of Token Passing Algorithm

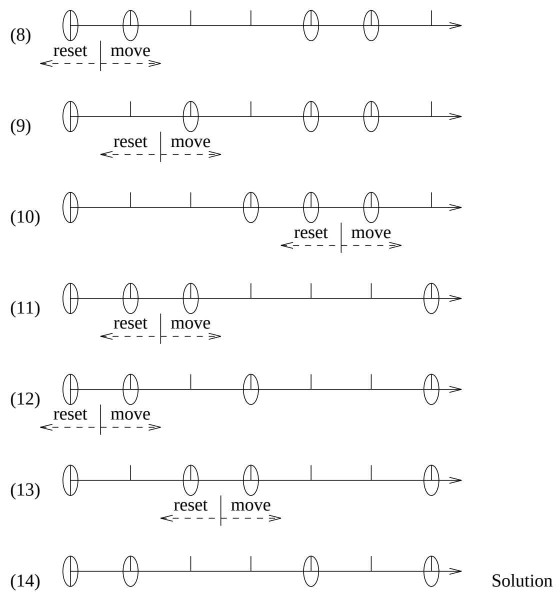

Figure 8: Example of Token Passing Algorithm (cont.)

anchored at position zero and never moves from it. The current mark is checked to see if it can be incremented. If it can then the position is incremented, all marks less than the current mark are reset to their initial positions, the TOKEN position is set to the second mark and the new ruler is checked with Diffncheck.

Since the TOKEN always starts from the second mark and tries to move its present mark as far to the right as it will go before it is passed to the next mark, we can ensure that no mark is incremented before all marks to its left have exhausted all possible positions. The shortcut of reseting the token to the second mark upon incrementation of any mark is equivalent to passing that token to the left upon incrementation, with the first mark passing it back to the second mark upon receiving it. Thus, a token received from the right means that it should be passed to the left, and a token that is received from the left means that the present position should be incremented (if possible) and the token passed to the left (to one of the reset marks), or, if not possible to increment the present mark position, the token should be passed to the right.

The result of this algorithm is a sequence of rulers where each mark in the ruler is incremented only after the lower marks have progressed through all possible combinations of positions. In this way, Iterate is guaranteed to generate all possible mark combinations. In addition, a completely distributed algorithm can be made with an iteration cost of $O(1)$ and a hardware cost of $O(N)$ cells for a N-mark Golomb ruler search. The resetting of all previous marks to their initial positions in reality has a time cost of $O(\ln N)$ if we assume limited fan-in gates, but since the number of marks in every practical case for the time being is around twenty the delay introduced in reset will not exceed two gate stages (even with fan-in of 5).

The Diffncheck testing procedure takes the set of marks and generates each row of the difference triangle. After the distances in one row are computed, they are individually compared with the contents of the difference triangle. If two distances are found to be the same, then the procedure immediately exits with a failed status. Otherwise, the procedure exits after generating and comparing all the individual values in the difference triangle. In a hardware implementation, the first order differences can be generated in parallel and pipeline stages can be used to form each order differences. Then, the differences can be also sorted in pipelined fashion with an equality of any two elements of the sort resulting in a negative outcome (the

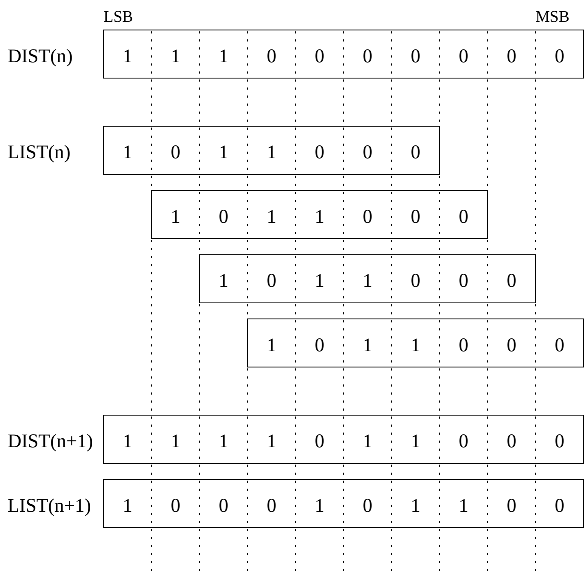

Figure 9: Example of Shift Algorithm

common case) and abort of the procedure. The cost of the sort network can be expensive, with hardware requirements of $O(k^2)$ or $O(k \ln k)$ depending on the algorithm used. In reality the sorting network can even be simplified to $O(1)$ hardware resources by taking the corresponding $O(k^2)$ time penalty, but this case is not of interest if fast generation of rulers is desired. The hardware cost of the sorting network, combined with the lack of use of previously generated information led to the characterization of the token passing algorithm but not its realization in hardware.

The iteration process is illustrated in Figures 7 and 8 for the four mark ruler case. The ovals represent marks on the ruler, with position zero being the leftmost mark. For each cycle through the algorithm, a single mark is moved, and all marks to the left of that mark are reset to their initial positions. The lowest mark is always held at the zero position. For example purposes, the initial position for each mark N is at the $N - 1$ location. In the actual implementation of the algorithm, these marks would be reset to their corresponding Golomb Ruler lengths, as described earlier.

This algorithm has the advantage that it will always produce the shortest ruler for a given number of marks first. It does not require an upper bound on the search space like the Sci. Am. and Shift algorithms do. However, the big disadvantage of this algorithm is in the use of the distance generation algorithm for ruler detection, as described above.

2.3 Shift Algorithm

This algorithm was developed by D. McCracken at Duke University. The essence of the algorithm is taken from the Sci. Am. Algorithm, but with major improvements in the area of ruler checking.

The first improvement comes from the fact that if the array of distances is preserved between computations, then when a new mark is added only the new distances created by that mark need to be added and checked against the array. So instead of $N(N-1)/2$ distances to compute, only N distances need to be computed.

The next improvement reduces the checking and generation overhead even further. The distance array is stored as a bitmap, rather than as an array of boolean values. Each bit

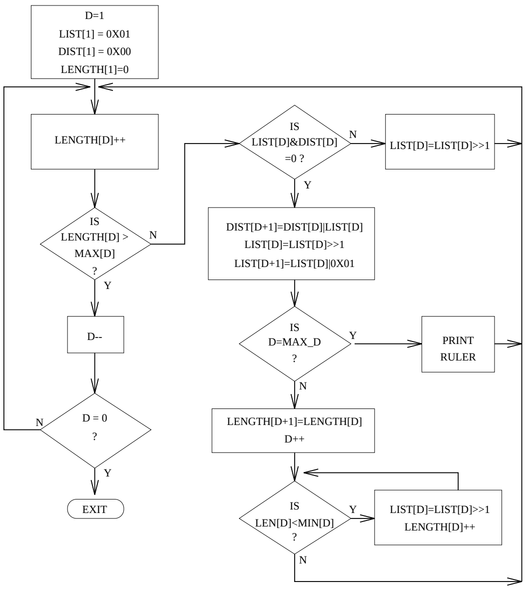

Figure 10: Flowchart for Shift Algorithm

position in this distance vector (which we will call $DIST$) represents a distance measured by the current ruler. The list of marks is also stored in a bitmap vector, called $LIST$. Each set bit in $LIST$ represents not so much the position of a mark, but rather the distance from each mark to a single non-existent point one position past the last mark of the current ruler. So $LIST$ can be pictured as the mirror image (shifted by one bit) of the ruler currently being evaluated.

In order to add a new mark to the ruler, the $LIST$ vector is shifted out until the distances in that vector do not intersect any distances already measured in the $DIST$ vector. This check is accomplished by simply AND-ing the two vectors together and checking to make sure that the resulting bit values are all zeros. In order to add the new point and all of its associated distances, the contents of $LIST$ are OR-ed into $DIST$ and the then lowest bit of $LIST$ is set to add the new point. These improvements reduce the computational complexity to add and check a new point from $O(N)$ to $O(1)$.

An example of this process is shown in Figure 9. The initial contents of the $LIST$ vector are for the three mark ruler $0, 1, 3$. The $LIST$ vector is shifted until it no longer has any set bits in common with those of the $DIST$ vector. At this point the current $LIST$ and $DIST$ vectors are saved and the contents of the new $DIST$ vector are updated by OR-ing in the contents of $LIST$, and the new mark is added to $LIST$ shifting the vector and setting the first bit. The end result is that $LIST$ now contains a representation of the four mark ruler $0, 1, 3, 7$ and $DIST$ contains the set of distances, $1, 2, 3, 4, 6, 7$, measured by this ruler.

When a valid $N + 1$ mark ruler is formed, the existing N mark ruler is pushed onto a stack, and the algorithm attempts to add another mark to the new top ruler. If no more marks can be added to the current ruler, then the stack is popped and a new ruler is generated. The search space for a Golomb Ruler can thus be modeled as a large tree structure. The nodes at level N represent valid N mark rulers. The children of that node represent the set of all possible Golomb Rulers that can be formed by appending a single mark to the set of marks in the parent ruler. The Shift Algorithm traverses this search space in a depth-first manner, utilizing the stack to provide backtracking. Discovering a ruler is simply a case of finding a node that exists at the desired depth in the tree.

A flow chart of the algorithm is shown in Figure 10.

In private communication with A. Dollas, O. Sibert indicated that he uses a similar algorithm, developed independently by him and yet unpublished.

2.4 Tree Algorithm

As discussed earlier, one of the problems with the Sci. Am. and Shift algorithms is the need to establish an upper bound on the search space prior to searching. If this upper bound is too low, then the Golomb Ruler will never be found. If this bound is too high, much time will be wasted in searching this extra space.

One algorithm that attempts to avoid this problem is the Tree Algorithm. The search space is represented as a general tree structure, with nodes representing marks and the edges representing distances. Finding a valid mark consists of locating a node at the proper depth. The tree is built up by applying the Shift Algorithm to each of the nodes of the tree, checking to see if new nodes can be added.

The outer loop of the algorithm sets the maximum ruler length to a minimum known value. A subroutine is then called which adds nodes to the tree structure. If a ruler is not found at the desired depth in the tree, the subroutine exits. The outer loop then increments the maximum length and calls the subroutine again, which adds more nodes to the existing tree. Since the data in the tree is preserved between calls to the subroutine, this algorithm can reuse the existing data to generate only the sections of the search space that have not been traversed previously.

The advantages of this approach is that the process is fairly fast. Also, this algorithm will find the minimum length ruler without having a predetermined maximum bound on the search space.

However, the big disadvantage is the size of the data structures needed to hold the tree is proportional to the search space, which grows geometrically with each additional mark. Use of this algorithm to find rulers greater that about ten marks is prohibitive due to its memory requirements.

3 Search Space Reduction

As pointed out earlier, the Shift Algorithm is as efficient as the authors know in the generation and checking of rulers. In order to improve the execution time, the search space must be reduced. There are a number of restrictions that can be made that will drastically reduce this space.

Search space reduction techniques involve "preclusion" or "branch pruning" of the search tree in order to eliminate as many false branches as early as possible in the search algorithm [5]. Effectively, all the preclusion techniques used here involve "bounding" the positions of the marks by establishing upper and lower limits for each mark.

The source of these techniques is cited when published. Some techniques are thought to have been previously used but not published yet.

3.1 First Mark Preclusion

One of the earliest published uses of preclusion techniques is given for the Sci. Am. Algorithm [3]. By starting the second mark at the distance 2 instead of 1, the search space could be reduced by about ten percent for shorter rulers, while still maintaining an exhaustive search. This technique relies on the fact that if a ruler has its second mark placed at the first position (measuring a distance of 1 unit), then the ruler's mirror image cannot have a mark also at the first position. The algorithm may not detect the first ruler, if indeed it has a mark at distance 1, but is still guaranteed to find the mirror image of that ruler.

3.2 Midpoint Reduction

A variation on this method relies on the fact that the mirror image of a ruler still measures the same distances, as we stated earlier. Therefore, we can limit the position of the middle mark of the ruler to one side of the center. We are still guaranteed to find a ruler, or the corresponding mirror image, because one of the two must have its middle mark before the geometric center of the ruler. This one restriction effectively cuts the search space in half [4]. Note, however, that for rulers with even numbers of marks, this technique does not exclude the possibility of "discovering" both rulers of a mirrored pair. The reduction factor for even mark rulers is thus less than for the odd mark case. Note also that this technique cannot be combined with the First Mark Preclusion technique described in Section 3.1.

3.3 Maximum Position Reduction

The next space reduction technique relies on the fact that for any mark on the ruler, the remaining larger marks cannot occupy a space smaller than the corresponding Optimum Golomb Ruler for those marks. Thus there is a minimum distance that any mark can be from the end of the ruler, based upon the maximum ruler size and the mark position. This technique can be further improved by realizing that any distance measured by the first part of the ruler cannot be measured in the second part. If we keep a table of minimum ruler lengths as a function of their length and distances measured, the upper bounds for marks can be reduced even further. This table is referred to as the *CHOOSE* array.

3.4 Minimum Ruler Preclusion

This method is based on the fact that for a ruler with a given number of marks, certain distances have to be measured. For example, the minimum 11 mark ruler is of length 72, so we will limit the 11 mark of our search space to no less than 72. Now, all 11 mark rulers of length 72 or less measure the distance 1. Therefore, when attempting to append the 12 mark onto a length 72, 11 mark ruler, this mark cannot be placed at position 73. By keeping a table with this information, we can avoid these conditions when generating rulers. This table is referred to as the *TCHOOSE* array.

4 Existing Optimum Rulers

The set of known Optimum Golomb Rulers is limited. Optimum rulers of up to sixteen marks have been previously published. Table 1 summarizes these known rulers. Reflections, or "Mirror Images" of rulers are not included in the list.

5 Results

Having verified the correctness of the parallel implementation of the Shift algorithm, the program is run with ruler lengths for which the optimum solutions were not known. The results of these runs are presented in this section.

5.1 New Rulers

Optimum seventeen, eighteen and nineteen mark Golomb Rulers were generated in separate runs on different workstation clusters. The rulers are shown in Tables 2, 3 and 4.

In deriving the seventeen and eighteen mark rulers, however, the authors were not the first to computationally complete the optimality search. Both rulers were derived first by Mr. Olin

Marks	Ruler
3	0,1,3
4	0,1,4,6
5	0,1,4,9,11
	0,3,4,9,11
6	0,1,4,10,12,17
	0,1,4,10,15,17
	0,1,8,11,13,17
	0,1,8,12,14,17
7	0,1,4,10,18,23,25
	0,1,7,11,20,23,25
	0,2,3,10,16,21,25
	0,2,6,9,14,24,25
	0,3,4,12,18,23,25
8	0,1,4,9,15,22,32,34
9	0,1,5,12,25,27,35,41,44
10	0,1,6,10,23,26,34,41,53,55
11	0,1,4,13,28,33,47,54,64,70,72
	0,1,9,19,24,31,52,56,58,69,72
12	0,2,6,24,29,40,43,55,68,75,76,85
13	0,2,5,25,37,43,59,70,85,89,98,99,106
14	0,5,28,38,41,49,50,68,75,92,107,121,123,127
15	0,6,7,15,28,40,51,75,89,92,94,121,131,147,151
16	0,1,4,11,26,32,56,68,76,115,117,134,150,163,168,177

Table 1: Known Optimum Golomb Rulers

0	8	31	34	40	61	77	99	118	119	132	143	147	182	192	194	199
8	23	3	6	21	16	22	19	1	13	11	4	35	10	2	5	
31	26	9	27	37	38	41	20	14	24	15	39	45	12	7		
34	32	30	43	59	57	42	33	25	28	50	49	47	17			
40	53	46	65	78	58	55	44	29	63	60	51	52				
61	69	68	84	79	71	66	48	64	73	62	56					
77	91	87	85	92	82	70	83	74	75	67						
99	110	88	98	103	86	105	93	76	80							
118	111	101	109	107	121	115	95	81								
119	124	112	113	142	131	117	100									
132	135	116	148	152	133	122										
143	139	151	158	154	138											
147	174	161	160	159												
182	184	163	165													
192	186	168														
194	191															
199																

Table 2: Difference Triangle for Optimum 17 Mark Ruler

Sibert in April and June 1993, respectively [17]. Identical results were independently produced by the authors in August and September 1993. O. Sibert's results have not been published to date and were relayed to the authors by Prof. Golomb. The nineteen mark ruler case was proven optimum by the authors in January 1994.

5.2 Computational Derivation of 17, 18 and 19 Mark Rulers

A variation of the Shift algorithm was developed using the PVM [8] distributed computing environment. A simple master/slave coding model was used. A single master process would generate short (5 to 6 mark) rulers that would be sent to one of a number of slave processes. Each of the slave processes would continue traversing the search space using the shorter ruler as a basis. When processing of shorter, *basis* ruler was complete, the slave would send a message back to the master, requesting another ruler.

Checkpointing was implemented within the master process to guarantee that the program could be restarted with no loss in the search space checking and without significant loss of previous work. The slave process granularity was set to a few hours of CPU time so that the overhead for starting and terminating processes would be small when compared to their CPU time, but the loss of any process would not result in a major loss of work. In addition, in the event that a slave process aborted, the work performed by that slave would be reissued to another slave. This guaranteed that the entire ruler search space was traversed and the search was exhaustive.

The code was run on a cluster of 12 Sun Sparc Classic workstations, and a single SparcServer 1000 fileserver. Since the cluster was utilized by several classes within the Engineering Department, the processes were run in the background with a low associated priority. Even with this limitation, it is estimated that the programs received 85-95% of the available CPU cycles over the course of several months. Because of the variable CPU load present over the course of the semester classes, it proved impractical to attempt to accurately measure the total amount of CPU time consumed by the search process.

The program was run over the course of several months from June 1993 through January 1994. During this time, the optimal of the 17, 18 and 19 mark rulers were determined. The 17 mark case used approximately 1440 CPU hours. The 18 mark case took approximately 8600

0	11	24	28	49	63	68	86	118	127	133	134	160	163	194	206	214	216
11	13	4	21	14	5	18	32	9	6	1	26	3	31	12	8	2	
24	17	25	35	19	23	50	41	15	7	27	29	34	43	20	10		
28	38	39	40	37	55	59	47	16	33	30	60	46	51	22			
49	52	44	58	69	64	65	48	42	36	61	72	54	53				
63	57	62	90	78	70	66	74	45	67	73	80	56					
68	75	94	99	84	71	92	77	76	79	81	82						
86	107	103	105	85	97	95	108	88	87	83							
118	116	109	106	111	100	126	120	96	89								
127	122	110	132	114	131	138	128	98									
133	123	136	135	145	143	146	130										
134	149	139	166	157	151	148											
160	152	170	178	165	153												
163	183	182	186	167													
194	195	190	188														
206	203	192															
214	205																
216																	

Table 3: Difference Triangle for Optimum 18 Mark Ruler

0	4	13	15	42	56	59	77	93	116	126	138	146	174	214	221	240	245	246
4	9	2	27	14	3	18	16	23	10	12	8	28	40	7	19	5	1	
13	11	29	41	17	21	34	39	33	22	20	36	68	47	26	24	6		
15	38	43	44	35	37	57	49	45	30	48	76	75	66	31	25			
42	52	46	62	51	60	67	61	53	58	88	83	94	71	32				
56	55	64	78	74	70	79	69	81	98	95	102	99	72					
59	73	80	101	84	82	87	97	121	105	114	107	100						
77	89	103	111	96	90	115	137	128	124	119	108							
93	112	113	123	104	118	155	144	147	129	120								
116	122	125	131	132	158	162	163	152	130									
126	134	133	159	172	165	181	168	153										
138	142	161	199	179	184	186	169											
146	170	201	206	198	189	187												
174	210	208	225	203	190													
214	217	227	230	204														
221	236	232	231															
240	241	233																
245	242																	
246																		

Table 4: Difference Triangle for Optimum 19 Mark Ruler

hours of CPU time to complete and the 19 mark case took approximately 36,200 CPU hours. All CPU times were normalized to a Sun Sparc Classic workstation.

6 Conclusions

The parallel implementation of the Shift Algorithm has been successful in identifying three previously unpublished Optimum Golomb Rulers, and is currently searching for the 20 mark Golomb Ruler.

An important research result is the demonstration of reliable parallel computation for extended execution times. Fault tolerance and restartability issues are addressed and have been shown to be effective in assuring processing reliability for program runs in excess of a month. Finally, previously unpublished results were found and verified against independent findings from another researcher, Mr. Olin Sibert.

7 Acknowledgements

The authors wish to thank the Duke University Department of Electrical Engineering for providing resources and support for this research. In addition, we would like to extend our appreciation to Emily Angerer for her help in implementation of the parallel algorithm and to Aviral Singh for his help at the early stages of the parallel runs.

References

[1] W. C. Babcock. Intermodulation Interference in Radio Systems. *Bell Systems Technical Journal*, pages 63–73, January 1953.

[2] Gary S. Bloom and Solomon W. Golomb. Applications of Numbered Unidirected Graphs. *Proceedings of the IEEE*, 65:562–570, April 1977.

[3] A. K. Dewdney. Computer Recreations. *Scientific American*, pages 16–26, December 1985.

[4] A. K. Dewdney. Computer Recreations. *Scientific American*, pages 14–21, March 1986.

[5] J Nievergelt E. M. Reingold and N. Deo. *Combinational Algorithms: Theory and Practice*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

[6] E. J. Blum F. Biraud and J. C. Ribes. On Optimal Sythetic Linear Arrays with Applications to Radioastronomy. *IEEE Transactions on Antennas and Propagation*, AP-22:108–109, 1974.

[7] M. Gardner. Mathematical Games. *Scientific American*, 226:108–112, March 1972.

[8] Geist, Beguelin, et. al. *PVM 3 User's Guide and Reference Manual*, May 1993.

[9] E. J. Kleiber. Some Difference Triangles for Constructing Self-Orthogonal Codes. *IEEE Transactions on Information Theory*, IT-16:237–238, 1969.

[10] Torleiv Klove. Bounds on the Size of Optimal Diffence Triangle Sets. *IEEE Transactions on Information Theory*, IT-34:355–361, 1988.

[11] Torleiv Klove. Bounds and Construction for Difference Triangle Sets. *IEEE Transactions on Information Theory*, IT-35:879–886, July 1989.

[12] N. Santoro M. D. Atkinson and J. Urrutia. Integer Sets with Distinct Sums and Differences and Carrier Frequency Assignments for Nonliner Repeaters. *IEEE Transactions on Communications*, COM-34:614–617, June 1986.

[13] Alan T. Moffet. Minimum-Redundancy Linear Arrays. *IEEE Transactions on Antennas and Propogation*, AP-16:172–175, March 1968.

[14] J. Robbins R. Gagliardi and H. Taylor. Acquisition Sequences in PPM Communications. *IEEE Transactions on Information Theory*, 33:738–744, Sept 1987.

[15] J. P. Robinson and A. J. Bernstein. A Class of Binary Recurrent Codes with Limited Error Propogation. *IEEE Transactions on Information Theory*, IT-13:106–113, January 1967.

[16] James B. Shearer. Some New Optimum Golomb Rulers. *IEEE Transactions on Information Theory*, IT-36:183–184, January 1990.

[17] W. Olin Sibert. In letter to H. Taylor and S. Golomb, April 1993.