GENETIC ALGORITHM APPROACH TO THE SEARCH FOR GOLOMB RULERS*

Stephen W. Soliday
soliday@garfield.ncat.edu

Abdollah Homaifar
homaifar@garfield.ncat.edu

Gary L. Lebby
lebby@garfield.ncat.edu

Department of Electrical Engineering
North Carolina A&T State University
Greensboro, North Carolina 27411

Abstract

The success of genetic algorithm in finding relatively good solutions to NP-complete problems such as the traveling salesman problem and job-shop scheduling problem provided a good starting point for a machine intelligent method of finding Golomb Rulers. These rulers have been applied to radio astronomy, X-ray crystallography, circuit layout and geographical mapping. Currently the shortest lengths of the first sixteen rulers are known. The nature of NP-complete makes the search for higher order rulers difficult and very time consuming. While the shortest lengths for each order are important as a mathematical exercise, finding relatively short high order valid rulers has a more important impact on real world applications. Genetic algorithm has shown good results in finding usable Golomb Rulers in minutes or hours instead of weeks or months.

1 INTRODUCTION

Golomb Rulers are a class of undirected graphs that were first described by Solomon W. Golomb, Professor of Mathematics and Electrical Engineering at the University of Southern California. The Golomb Ruler measures more discrete lengths than the number of marks it carries. It does not measure the same distance twice (Bloom, Golomb 1977). For example, a normal 12 inch ruler contains 13 marks and is capable of measuring 12 discrete lengths. Each measurement is made by summing the lengths of the segments between two marks. There are 78 combinations of two marks in the 13 mark ruler. Of these 78 combinations only 12 of them are unique measurements, and the other 66 are redundant. By making the lengths of the segments of the ruler different from one another, it is possible to make more unique measurements than the number of marks on the ruler. For most permutations of the segments of these rulers, there still exist measurements that are redundant. The few permutations that produce rulers with no repeats in measurements are known as Golomb Rulers.

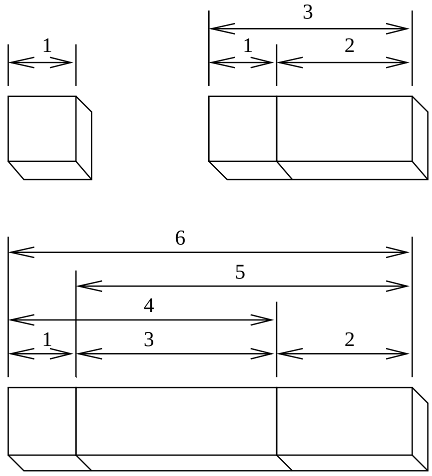

Figure 1: Three Perfect Golomb Rulers

For a ruler to be a perfect Golomb Ruler it must meet the following criteria. First, it must be constructed of segments of unique integral length. In other words, the length of a segment must be some integer number of units and no two segments in a given ruler may have the same length. Second, the ruler must be able to measure discrete spans. That means that the distance between two given marks must be a unique length for that ruler, with no two spans equaling the same length.

*This paper was received for review in Jan 1995. Subsequently, it was accepted and published in the *Proceedings of the Sixth International Conference on Genetic Algorithms*, Vol 1., pp 528–535, Morgan Kaufmann, University of Pittsburgh, PA, Jul 1995, ISBN: 1–55860–370–0

Finally, for the class of rulers known as perfect rulers, all the intervals between one and the overall length of the ruler must be measured without skipping a measurement. Unfortunately there are only three perfect rulers. Written in the mark representation, the three rulers in Figure 1 are:

```
n = 2    0, 1
    3    0, 1, 3
    4    0, 1, 4, 6
```

The numbers correspond to the unit positions of the marks on the ruler. These three rulers are capable of measuring from 1 to their overall length without repeating or skipping an interval. Note that if we were to permute the $n = 4$ ruler from $(0, 1, 4, 6)$ to $(0, 1, 3, 6)$ it would no longer be a Golomb Ruler. This permuted ruler, would measure the intervals $(1, 2, 3, 3, 5, 6)$. Notice that the span of length 3 is measured twice and the span of length 4 is never measured. Figure 1 shows the only perfect rulers (Bloom, Golomb 1977).

If there only exist three perfect Golomb Rulers then a new definition must be made. More to the point, one of the constraints of the perfect Golomb Ruler must be relaxed. The only constraint that may be relaxed and still leave us with a useful tool is the one that states that none of the distances between 1 and the overall length may be skipped. Given this leniency on the skipping of intervals, it may be suggested that there would be an infinite number of $n = 5$ rulers. While this is true, the shortest ruler possible with $n = 5$ is what is desired (Carter, Robertson, et.al. 1984).

Golomb Rulers represent a unique class of NP–complete problems. Unlike the Traveling Salesman problem (TSP), which may be classified as a "*complete ordered set*", the Golomb Ruler may be classified as an "*incomplete ordered set*". The optimum solution for the latter is a single permutation of a set of m elements taken n at a time, where $n \ll m$.

Arthur C. Clarke, in a discussion about a bid for an engineering project, wrote (Clarke 1990):

"How long will it take?"
"Do you want it quick, cheap, or good?
I can give you any two."
"Fairly quick and very good..."

In an ideal world we want the best solution to a problem. However, in the real world of budgets and deadlines we find that the choice is usually to do things fast and cheap. Here is where a goal of optimization comes in. Given the constraints of time and of dollars we cannot always find the perfect solution to a problem, and so we must find good solutions. The focus of this paper is to find high order *good* Golomb Rulers. To date the highest order ruler whose shortest length is known is the ruler with 16 marks (Robertson, 1994). Rulers of this order would require years of exhaustive searching, and weeks and sometimes months if heuristics are employed. See Table 1. In this paper a genetic algorithm (GA) is used to evolve good rulers of order 5 to 16. The GAs were executed on a desktop computer and the execution times were between 1 second and 26 minutes each.

Golomb Rulers have been applied to radio astronomy, X-ray crystallography, circuit layout and geographical mapping (Bloom, Golomb 1977; Carter, Robertson et. al. 1984). One of the researchers whose work is directly affected by the solution of Golomb-Rulers is a Geophysicist at the National Oceanic and Atmospheric Administration in Rockville, MD. Using VLBI (*Very–Long Baseline Interferometry*) he is engaged in finely detailed measurements of the earth. In base line interferometry, phase difference measurements are made between two separated telescopes. This is then repeated for different separation distances. Using Fourier analysis, the various phase differences may be used to reconstruct the total phase difference and thereby recover the angle between the radio source and the baseline, with a high degree of resolution. If the separation distance between two pairs of antennas is not significantly unique, then the measurements are redundant. By placing the antennas on the marks of a large Golomb ruler, the researcher is assured that no two measurements will be redundant (Carter, Robertson et. al. 1984). Remember that with the 13 mark inch ruler, out of 78 pairs of measurements only 12 were unique. When the cost of observing time is considered, 12 percent efficiency is rather undesirable.

2 EXHAUSTIVE SEARCH WITH HEURISTICS

2.1 NP-COMPLETE

Golomb Rulers represent a class of problems known as NP-complete. The exhaustive search, without heuristics, of such problems is impossible for higher order models (Soliday 1990). As another mark is added to the ruler, the time required to search the permutations and to test the ruler becomes exponentially greater.

Earlier it was stated that the Golomb Ruler is an *incomplete ordered set*. Golomb Rulers are classified by their order (n), where n indicates the number of marks. For a ruler this means that there are $n - 1$ segments. For a given order n and a maximum segment length m the number of permutations, or possible rulers that may be formed is:

$$\begin{aligned} \text{number of rulers} &= \tfrac{1}{2}\binom{m}{n-1} \\ &= \tfrac{m!}{2(m-n+1)!} \end{aligned} \tag{1}$$

The 2 in the denominator, of Equation 1, reflects that fact that mirror images do not produce unique rulers. The number of tests made on an order n ruler in order to determine redundancy of measurement and thereby check the validity of the ruler, is:

$$\begin{aligned} \text{number of spans} &= \textstyle\sum_{i=1}^{n-1} i \\ &= \tfrac{n(n-1)}{2} \end{aligned} \tag{2}$$

Let us imagine for a moment that we possessed a hypothetical computer able to measure and compare the discrete intervals of our Golomb Rulers at a fictitious speed of one measurement and test every nanosecond (10^{-9} seconds). How long would it take to exhaustively search for the shortest n mark ruler with a maximum individual segment length of m?

$$\text{search time} = \frac{n\,(n-1)\,m!}{4(m-n+1)!}10^{-9}\text{sec} \tag{3}$$

Table 1 provides some of the times that would be required for an exhaustive search with no heuristics. Note that the last entry represents $4.6x10^{10}$ times the age of the known universe (Kaufmann 1985).

Table 1: Execution times for non–heuristic exhaustive searches

n	m	time	units
6	7	1.89e-05	sec
8	13	0.121	sec
10	19	12.57	min
12	30	2.28	years
14	36	2.07e+04	years
16	49	3.92e+09	years
18	65	1.61e+15	years
20	83	9.36e+20	years

2.2 HEURISTICS

Clearly it can be seen that exhaustively searching for the shortest Golomb Rulers is impractical and sometimes impossible. The search space may be drastically reduced if heuristics are used (Dewdney 1985). At each stage of permutation generation and ruler construction, tests are performed. The principle method of testing a Golomb Ruler is to maintain a history list of measurements. As each pair of marks is tested the span between them is compared to the history list. A match would invalidate the ruler. The principle *time cutting* theorem states that "*any piece of a Golomb Ruler must itself be a Golomb Ruler*" (Bloom, Golomb 1977). As soon as a piece of a ruler is invalidated, all permutations of the unchecked segments may be skipped.

3 USING GENETIC ALGORITHM TO SEARCH FOR GOLOMB RULERS

The success of genetic algorithm in finding relatively good solutions to NP–complete problems such as the traveling salesman problem (Goldberg, Lingle 1985; Oliver, et.al. 1987; Grefenstette, et.al. 1985; Jog, et.al. 1989; Liepins, et.al. 1990; Whitley, et.al. 1989; Lid 1991; Homaifar, et.al. 1992,1993) and job–shop scheduling (Liepins, et.al. 1987; Davis, Smith 1985; Whitley, Starkweather 1989; Cleveland, Smith 1989) provided a good starting point for a machine intelligent method of finding *good* Golomb Rulers. The theorem that provides a means of removing unnecessary permutations in the heuristic–based approach also lends a hand in the GA approach. The theorem stated that "*any piece of a Golomb ruler is itself a valid ruler*". This is essentially the building block hypothesis (Goldberg 1989). When the GA begins, the rulers will contain low order Golomb Rulers within themselves. As the population approaches the optimum point, the rulers will contain higher order Golomb rulers.

3.1 REPRESENTATION

Typically Golomb Rulers are represented by the position of the marks on the ruler. Using this method created problems with developing a good crossover operator. It became clear early on that the Golomb Ruler represented permutations of an *incomplete ordered set*. The representation used in this program is one in which the length of each segment is represented by the element of an integer array. The position of a segment on the ruler corresponds to an index for the array (*if ruler*$(5) = 7$, *then the fifth segment on the ruler has a length of seven.*)

In order to assure unique segment lengths, the first representation tried contained not one but two lists of ruler segments. The first list was the *ruler* and the second list was the *surplus*. Mutation was performed by either swapping ruler segments with each other, thereby creating a new permutation, or by swapping a ruler segment with a surplus segment.

By introducing multiple fitness criteria, a simpler representation replaced the two list method. In this paper only the *ruler* segments are maintained in the list. This effectively reduced the search space but required modifications to the simple GA. Each of the modifications will be discussed in the separate sections below.

3.2 INITIAL POPULATION

The initial population was chosen at random. This was accomplished by loading an array with the numbers 1 to m. Here the original concept of using a *ruler* and a *surplus* was maintained. The positions 2 thru m were then scrambled. Scrambling consists of randomly selecting two positions in the array i and j with $i \neq j$. The values at the two positions are then swapped $A_i \rightleftharpoons A_j$. This random swapping was done $m/2$ times. The purpose of only swapping the elements 2 thru m is to assure that the segment length 1 is retained in the *ruler* portion of the list. The next step is to copy the first $n-1$ elements into the ruler array. Next the ruler itself is randomized by using the same swapping method, this time including the segment length 1. Finally the ruler is aligned to prevent mirror image representations. See Section 3.6

3.3 EVALUATION

There are two fitness criteria of a ruler with unique segment lengths. The first criterion is the overall length of the ruler, and the second criterion is the number of repeated measurements. The second fitness criterion is the most important: if and only if there are no repeated measurements is the ruler a Golomb Ruler.

3.3.1 SCORING THE RULERS

A histogram of lengths is built by measuring the spans of all combinations of adjacent segments. The sum of the segments is the overall length. All values in the histogram are decremented by one, anything less than zero was considered to be zero. The sum of the histogram then becomes the number of repeated measurements. These two values were used along with a unique list of coefficients to produce a fitness function. The goal of the GA was to maximize the fitness function.

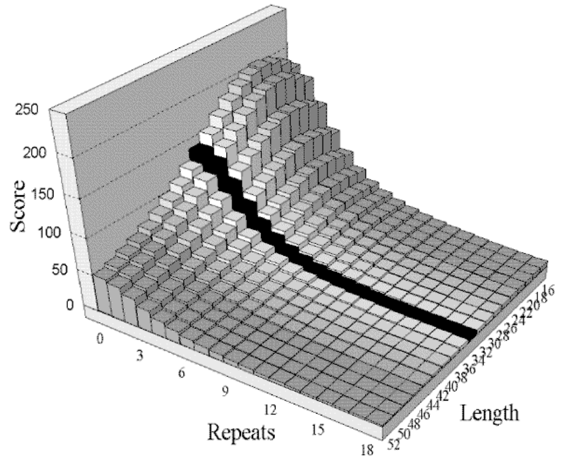

Figure 2: Comparison of (Repeats) vs. (Length)

3.3.2 FITNESS FUNCTION

Traditionally GAs are used to maximize fitness functions. However, the two fitness parameters in this application must be minimized (***repeats must go to zero, and overall length should be a small as possible.***) The fitness score of an individual should rise exponentially as it approaches the optimum point. To accomplish this a constant scale value was divided by the product of two polynomials. Figure 2 shows the surface created by the following equation.

$$\text{fitness} = \frac{\text{scale}}{(W_{r,0}+W_{r,1}R+W_{r,2}R^2)} \cdot \frac{1}{\left(W_{l,0}+W_{l,1}(L-W_{l,3})+W_{l,2}(L-W_{l,3})^2\right)} \tag{4}$$

Here the subscripts R and L represent the number of repeats in measurement and the overall length. The highest point on the dark band in Figure 2 represents the optimum point for an order 8 ruler. Figure 3 illustrates the ranking of fitness surrounding the optimum point (*higher numbers are better*) Using a spreadsheet that displayed both Figure 2 and a linear representation of Figure 3, the scale and the six weights were manually adjusted. The scale was used only to put the average fitness at a reasonable number. The fitness at the optimum point was arbitrarily placed around 100. By altering the weights to visually provide a smooth gradient that was steeper for the number of repeats than for the length, good fitness functions could be produced for each order ruler. The constraints set by Figure 3 force the GA to favor lower repeats over shorter lengths. This is also accomplished by making the gradient in the direction of lowering repeats greater than in the direction of decreasing overall length.

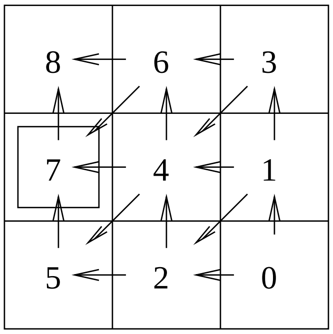

Figure 3: Rank of scores surrounding the optimum point

3.4 MUTATION

The mutation operator consisted of two types of mutation. A mutation could result in either a permutation in the segment order, or a change in the segment lengths. Mutation was handled individually for each segment in the ruler. For each ruler segment a random number was generated and this was compared to the probability that mutation will occur (P_m). If the probability was satisfied, a new random number was generated. This number decided whether the mutation would result in a permutation of segment order or if the mutation would result in changing the segment's length (P_x).

If it was determined that the mutation should be in the form of a permutation, then another segment would be chosen at random and the positions of the two segments would be swapped. However, if the mutation should produce a change in segment length then an integer number was repeatedly chosen at random between 2 and m until a number was found that was not currently being used in the ruler. The segment was replaced with this new number. This two step mutation was carried out for each segment in the ruler. The segment with the length of 1 was restricted to the first type of mutation.

Because of the strict constraints imposed by the definition, Golomb rulers are highly susceptible to destruction under mutation. Therefore a copy of the ruler is made before each mutation and if the mutation results in a lower fitness score, then the old copy is used to restore the ruler to its original values.

The original representation used employed a surplus list. In that representation only the ruler portion was mutated. If the mutation resulted in a change in segment length then a segment from the surplus was chosen at random and swapped with the segment from the ruler list.

3.5 SELECTION

The selection method used in this GA was tournament selection. A variable number of individual rulers was selected from the population at random. The scores of the individuals were compared and the highest scoring ruler was retained as parent number one (P_1) for the crossover operation. The process was repeated a second time to obtain the second parent (P_2). After the crossover was performed the parents were returned to the old population list and the children were placed in the new population list. This was repeated until the new population list was full.

3.6 CROSSOVER

Once the two parents were selected, it remained to be determined if crossover should occur. A random number was rolled and compared to the probability of crossover (P_c). If crossover was not performed, then the two children were just clones of the two parents. If crossover was performed, it was done in a fashion similar to partially mapped two point crossover.

Before the crossover was carried out the two parents were aligned with each other to reduce the chance of crossing two mirror images and thus destroying a possible good ruler. Since it was determined that all good rulers should have a segment of length one (Bloom, Golomb 1977), this became the tool for alignment. If the unit one segment was right of the center, the ruler's order was reversed.

The procedure for the crossover can be best described by following an example through the operation. P_1 and P_2 are the selected parents and C_1 and C_2 are the children.

P1:	2	1	5	4	6	3	7	8
P2:	4	5	7	6	8	1	2	3

Alignment reverses the order of parent two:

P1:	2	1	5	4	6	3	7	8
P2:	3	2	1	8	6	7	5	4

Cut points are randomly determined:

P1:	2	1	/	5	4	6	/	3	7	8
P2:	3	2	/	1	8	6	/	7	5	4

The segments between the cut points are marked in the opposite parent:

P1:	2	**1**	/	5	4	**6**	/	3	7	**8**
P2:	3	2	/	1	8	**6**	/	7	**5**	**4**

Table 2: Results of the Golomb Ruler Genetic Algorithm

O	LB	Sh	Pop	Gen	Time	Len	Rel Err	Representation
5		11	512	10	0 0.05	11	0%	2,5,1,3
6		17	512	22	0 0.15	17	0	1,7,3,2,4
7		25	512	22	0 0.17	25	0	1,6,4,9,3,2
8		34	1024	84	0:13	35	2.94	4,1,12,7,3,6,2
9		44	1024	433	1:22	44	0	1,4,7,13,2,8,6,3
10		55	1024	458	1:43	62	12.73	1,10,5,8,17,12,2,4,3
11		72	1024	152	0:39	79	9.72	2,5,1,22,11,9,12,4,10,3
12		85	1024	59	0:18	103	21.18	2,5,1,27,17,4,11,14,9,3,10
13		106	1024	664	4:03	124	16.98	1,5,19,23,9,22,12,16,2,8,3,4
14	114	127	2048	1506	21:38	168	32.28	6,9,1,4,7,17,32,19,18,22,8,23,2
15	133	151	2048	858	14:34	206	36.42	10,1,14,3,2,24,27,31,33,4,12,23,13,9
16	154	177	2048	708	26:29	238	36.46	1,6,3,13,5,29,30,11,5,39,42,2,17,24,8,4

The first child is created by copying the segments outside the cut points:

P1:	2	**1**	/	5	4	**6**	/	3	7	**8**
C1:	2	1	/	-	-	-	/	3	7	8

Next the segments between the cut points are replaced using the order they appear in the second parent:

C1:	2	1	/	6	5	4	/	3	7	8

The same procedure is used to create the second child. In this case the roles of the first and second parent are reversed. The full crossover is shown with both parents and both children.

P1:	2	1	5	4	6	3	7	8
P2:	3	2	1	8	6	7	5	4
C1:	2	1	6	5	4	3	7	8
C2:	3	2	1	6	8	7	5	4

In the original representation both the ruler list and the surplus list were used in the crossover. The partially mapped crossover was performed in the same way. The alignment differed in the method of aligning the surplus lists. The ruler lists were still aligned to prevent crossing mirror images. The surplus list did not benefit however from having a known segment length that could be shifted to the left of the center. Instead a Mean Square Difference (MSD) was found by summing the squares of the difference of the elements in the two parents' surplus lists. Next the second parent's list was reversed and a second MSD was determined. The order of the second parent's list that produced the lower MSD was kept.

4 RESULTS

The performance of GA is shown in Table 2; the column denoted by the heading (**O**) indicates the order of the ruler. The order of the ruler represents the number of marks contained on the ruler. The columns (**LB**) and (**Sh**) represent the computed *Lower Bound* and the *Shortest Known* rulers of a given order(Robertson, 1994). If a lower bound is not given, then the length reported in (**Sh**) has been proven to be the shortest ruler. Columns (**Pop**) and (**Gen**) report the initial population size and the number of generations required to produce the reported length. The (**Time**) column reports the execution time for the number of generations listed, measured in minutes and seconds. All runs were made to a maximum number of generations provided by the user that was higher than the listed generations form Table 2. The (**Len**) and (**Rel Err**) columns report the shortest length produced during the run of the GA and the relative error of that length with the shortest known length. Finally the (**REP**) column shows the segment length representation of the evolved Golomb Rulers.

The GA in this paper was written in C++ using object oriented programming. A GA class was written to be robust and to handle many types of GA. A population member class was designed to work with the GA class and yet not contain the knowledge of the GA operators—in this way the operations of GA remain general. To make a specific type of GA, a class was created and was derived from the GA member. This class was given member functions for the various GA operators (*i.e. Crossover, Mutate, Evaluate, Randomize, etc.*) The software was written to be compiled under GNU C++ version 2.5.0 and higher. This GA was executed on a 60Mhz Pentium under the Linux operating system. The execution times varied from 0.05 seconds for the $n = 5$ ruler to 26 minutes for the $n = 16$ ruler. In an earlier program, one that employed a heuristic based exhaustive search, the times varied from 0.035 seconds for $n = 5$ to 6 weeks for $n = 13$.

The GA was able to evolve the shortest ruler in $n = 5$–7, 9. In $n = 8$ the GA evolved a ruler that was 1 unit longer than the shortest known. For $n = 10$–16 the GA was able to produce good rulers in under 30 minutes. The number of valid rulers, (*i.e. the rulers with no repeated measurements*), in the population was examined. Initially in the random population that initiated the GA there were only 15 valid rulers out of 1024 for the 8 mark ruler. This number grew rapidly and maintained a range of 65 to 75 percent of the population.

5 CONCLUSIONS

The purpose of machine intelligent optimization is not necessarily to produce perfect results, but to produce the best results under the constraints of time and cost. This GA was successful in producing *short* rulers for each of the orders.

5.1 FUTURE PROJECTS

Imagine a modification to the classic TSP, where in addition to the cost of moving along the edges, we assign a return for reaching the nodes. As in the classic description of TSP, the salesman expends a cost to travel from city to city, but now lets have him receive a payment for sales at each city. He has a client list of 100 cities but during this trip he can only visit 80 of them. Which cities does he go to, and in what order does he visit them, to maximize his sales and minimize his travel costs? This is another example of an "*incomplete ordered set*". The GA described in this paper should prove effective for solving such a problem.

Acknowledgments

The work of S. Soliday, G. Lebby, and A. Homaifar was supported in part by grants from the Advanced Research Projects Agency and NASA Center of Research Excellence under contract number N00600–93–K–2051 and grant number NAGW–2924, at North Carolina A&T State University. The authors wish to thank them for their support.

References

Gary S. Bloom and Solomon W. Golomb. Applications of numbered undirected graphs. *Proceedings of the IEEE*, 65(4):562–570, 1977.

Arthur C. Clarke. *Ghost From the Grand Banks.* Bantam Books, New York, 1990. pp 88.

W. E. Carter, D. S. Robertson, J. E. Pettey, B. D. Tapley, B. E. Shultz, R. J. Eanes, and Miao Lufeng. Variations in the rotation of the earth. *Science*, pages 957–961, 1984.

Gary A. Cleveland and Stephen F. Smith. Using genetic algorithms to schedule flow shop releases. In Schafer, editor, *Proceedings of the Third International Conference for Genetic Algorithms*, pages 160–169. Morgan Kaufmann Publishers, Inc., 1989.

A. K. Dewdney. Computer recreations. *Scientific American*, 253(6), Dec 1985.

L. Davis and D. Smith. Adaptive design for layout synthesis. internal report, Texas Instruments, Dallas, 1985.

J. J. Grefenstette, R. Gopal, R. Rosmaita, and D. V. Gucht Genetic algorithms for the traveling salesman problem. In Grefebstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 154–159. Texas Instrument and U.S. Navy Center for Applied Research and Artificial Intelligence, 1985.

David E. Goldberg. *Genetic Algorithms, in Search, Optimization & Machine Learning.* Addison–Wesley Publishing Company, Inc., Massachusetts, 1989.

David E. Goldberg and R. Lingle Alleles, loci and the traveling salesman problem. In Grefebstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms.* Lawrence Erlbaum Associates, 1985.

A. Homaifar, S. Guan, and G. E. Liepins Schema analysis of a new approach to the traveling salesman problem by genetic algorithm. *Complex Systems*, 6(6):533–552, 1992.

A. Homaifar, S. Guan, and G. E. Liepins A New Approach on the Traveling Salesman Problem by Genetic Algorithms. In Schafer, editor, *Proceedings of the Third International Conference for Genetic Algorithms*, pages 460–466. Morgan Kaufmann Publishers, Inc., 1989.

William J. Kaufmann. *Universe.* W. H. Freeman and Company, New York, 1985. pp. 521.

G. E. Liepins, M. R. Hilliard, and M. Morrow Greedy genetics. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 231–235. Lawrence Erlbaum Associates, 1987.

G. E. Liepins, M. R. Hilliard, J. Richrdson, and M. Palmer. Genetic algorithm applications to set covering and traveling salesman problems. In Brown and White, editors, *Operations Research and Artificial In-*

telligence: the Integration of Problem Solving Strategies, pages 29–57. Kluwer Academic Press, 1990.

Mark L. Lid. Traveling salesman problem domain application of a fundamentally new approach to utilizing genetic algorithms. Technical report, Air Force Office of Scientific Research and Office of Naval Research, 1991. Research sponsored by Contract F4920–90–G–0033.

I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference of Genetic Algorithms*, pages 224–230. Lawerence Erlbaum Associates, 1987.

P. Jog, J. Y. Suh, and D. VanGucht The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem. In Schaffer J. D., editor, *Proceedings of the Third International Conference for Genetic Algorithms*, pages 110–115. Morgan Kauffmann Publishers, Inc., 1989.

Stephen W. Soliday. The Search for Golomb Rulers, Testing the Computational Power of Todays Computers. paper presented at the annual zone meeting, Southeastern section of the American Physical Society, Atlanta, 1990.

D. Whitley and T. Starkweather. Genitor ii: A distributed genetic algorithm. *J. Expt. Thoer. Artif. Intell.*, 2:189–214, 1989.

D. Whitley, T. Starkweather, and D. Fuquay Scheduling problems and traveling salesman: the genetic edge re-combination operator. In Schafer, editor, *Proceedings of the Third International Conference for Genetic Algorithms*, pages 133–140. Morgan Kaufmann Publishers, Inc., 1989.