# Solving Facility Layout Problems Using Genetic Programming

**Jaime Garces-Perez, Dale A. Schoenefeld and Roger L. Wainwright**

Department of Mathematical and Computer Sciences
The University of Tulsa
600 South College Avenue
Tulsa, OK 74104-3189
email: [schoend,rogerw]@utulsa.edu
url: http://euler.utulsa.edu/~schoend

## Abstract

**This research applies techniques and tools from Genetic Programming (GP) to the facility layout problem. The facility layout problem (FLP) is an NP-complete combinatorial optimization problem that has applications to efficient facility design for manufacturing and service industries. A facility layout is represented as a collection of rectangular blocks using a slicing tree structure (STS). We use a multiple purpose genetic programming kernel to generate slicing trees that are converted into candidate solutions for an FLP. The utility of our techniques is established using eight previously published benchmark problems. Our genetic programming techniques that evolve STSs are more natural and more flexible than all of the previously published genetic algorithm and simulated annealing techniques. Previous genetic algorithm techniques use a two-phase optimization strategy. The first phase uses clustering techniques to determine a near optimal fixed tree structure that is represented as a chromosome in a genetic algorithm. Within the constraints implied by the fixed tree structure, genetic algorithm techniques are applied during the second phase to optimize the placement of facilities in relation to each other. Our genetic programming technique is a single phase global optimization strategy using an unconstrained tree structure. This yields superior results.**

## 1 Introduction

The problem of facility layout is to decide the proper positioning of a collection of facilities on a planar site. Each facility has a required area and there is an interconnection cost between each pair of facilities. The interconnection cost between any pair of facilities is a quantitative flow of "materials" cost that respects both the amount of material that must be moved and the distance between the two facilities. The facility layout problem is an NP-complete combinatorial optimization problem that has applications in many contexts including architectural space planning, manufacturing cell layout and VLSI design [Tam and Li, 1991].

Historically, the Facility Layout Problem (FLP) has been modeled as a quadratic assignment problem, a quadratic set covering problem, a linear integer programming problem, a mixed integer programming problem, and a graph theoretic problem. Kusiak and Heragu provide a comprehensive survey of the facility layout problem [Kusiak and Heragu, 1987].

More recently, FLP research has progressed in various ways. More sophisticated problem solving frameworks, including genetic algorithms (GA) and simulated annealing, have been applied [Tam, 1992a, Tam, 1992b, Smith and Tate, 1993, Kado, 1995]. A variety of layout representations have also been developed. One promising layout representation technique uses a Slicing Tree Structure (STS) due to Otten [Otten, 1982]. The STS is often a useful layout representation; however, there are some physical layouts that cannot be represented by an STS, and there are other layouts for which the STS is not suitable. An STS specifies a general facility layout topology that, after application of some procedure, determines a specific facility layout.

This research applies techniques and tools from the Genetic Programming (GP) problem solving framework to the facility layout problem. We represent a facility layout as a collection of rectangular blocks using the Slicing Tree Structure. We use a multiple purpose genetic programming kernel written by Fraser [Fraser, 1994] to evolve slicing tree structures that correspond to optimal or near optimal solutions to the FLP. Our solutions are competitive with previously published facility layouts for eight benchmark problems that are commonly found in the literature, and that were recently used by Kado [Kado, 1995] and by Tam and Li [Tam and Li, 1991]. Our solutions are superior to Kado's results for the two largest problems containing, respectively, 20 and 30 facilities. Our genetic programming techniques applied to STSs are more natural and more flexible than previously published genetic algorithm and simulated annealing techniques. Previously published genetic algorithm techniques use a two-phase optimization strategy. The first phase uses a clustering strategy to determine a near optimal fixed tree structure. The second phase represents the fixed tree structure resulting from phase one as a fixed length chromosome and, using this fixed tree structure, attempts to optimize the placement of facilities in relation to each other [Kado, 1995]. Our genetic programming is a single phase global optimization strategy that does not constrain the tree structure. We determine the facility layout corresponding to an STS by using a variation of the "bottom-up" interpretation used by Kado [Kado, 1995]. To the best of our knowledge, our research is the first to apply GP to the facility layout problem.

## 2 The FLP Problem

We use the following formulation of the Facility Layout Problem (FLP). Other variations of the FLP are reviewed by Tam and Li [Tam and Li, 1991].

There are $n$ two-dimensional rectangular facilities that must be positioned in a site area. The site area can be any subset of the Euclidean plane (i.e., we consider an FLP without room limitations.) The site area contains no obstructions (i.e., we can use any part of the Euclidean plane for any facility). Each facility, $F_i$, is modeled as a rectangular block having area, $A_i$, determined by the facility requirement. Only a horizontal or a vertical orientation of each rectangular block is acceptable. If a block is allowed to be placed either horizontally or vertically, it is a *free orientation* block; otherwise it is a *fixed orientation* block. The *aspect ratio*, $r_i$ of a block is defined as the ratio of the width of the block to its height. A block with a variable aspect ratio defined over a range of values is a *flexible* block; otherwise it is a *rigid* block. Each block can be placed in any location within the site area but blocks cannot overlap one another. Let $w_{i,j}$ represent the weight of the material flow and let $d_{i,j}$ represents the Euclidean distance from the center of *block* $i$ to *block* $j$. Our objective is to minimize

$$flow = 0.50\Sigma_{i<j}(w_{i,j}d_{i,j}^2).$$

Our objective function is the same objective function used by Tam and Li [Tam and Li, 1991] and by Kado [Kado, 1995]. Our objective function is the function resulting by insisting that there are no aspect ratio violations and by letting $a = 1$, $b = 2$, $P_a = 0.5$, $P_b = 10^6$, and $P_c = 0$ in the following generalized flow function presented by Kado [Kado, 1995]:

$$flow = \\ P_a\Sigma_{i<j}(w_{i,j}^a d_{i,j}^b) + P_b\Sigma_i(asp\_break)_i + P_c(total\_area).$$

In this more general formulation, the term $(asp\_break)_i$ is used to indicate the extent that the $i$-th facility violates the given aspect ratio limitation. The term $(total\_area)$ is the minimum rectangular area that encloses all facilities.

Another objective function, resulting from alternate parameter settings and frequently used in other research, is given by

$$flow = 2.00\Sigma_{i<j}(w_{i,j}d_{i,j}).$$

## 3 The STS Representation Technique

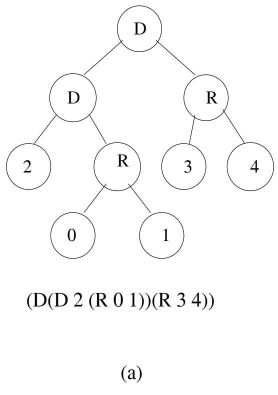
A Slicing Tree Structure (STS) for $n$ facilities is a binary expression tree where each operand (terminal) node is a label given by a facility index (0, 1, 2, ..., or $n$-1), and each operator (internal) node expresses the relation between its children substructures. The operator is U if the facility substructure represented by the left child is just Up from the facility substructure represented by the right child. Similarly, the operator is D, L, or R, respectively, if the facility substructure represented by the left child is just Down, just Left, or just Right, of the facility substructure represented by the right child. An STS is constrained to have exactly $n$ terminal nodes and each index must occur exactly once as a terminal node. It has been noted that different STSs may represent the same relationship among facilities [Wong and Liu, 1986, Kado, 1995]. For example, using the interpretation that we describe later, the expression tree illustrated in Figure 1(a) with symbolic expression given by

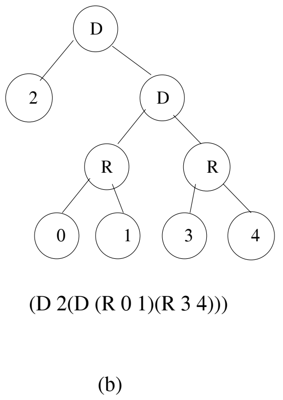
```
(D ( D 2 ( R 0 1 ) ) ( R 3 4 ) )
```

and the expression tree illustrated in Figure 1(b) with symbolic expression given by

```
( D 2 ( D ( R 0 1 ) ( R 3 4 ) ) )
```

will each determine the same initial facility layout shown in Figure 2. The redundancy is also increased by using four operators rather than two since, e.g., ( L 1 0 ) and ( R 0 1 ) represent the same substructure. Hence, the initial facility layout determined by the expression tree





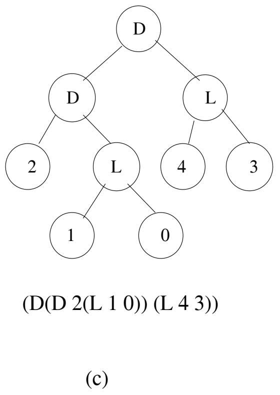


Figure 1: Equivalent Expressions

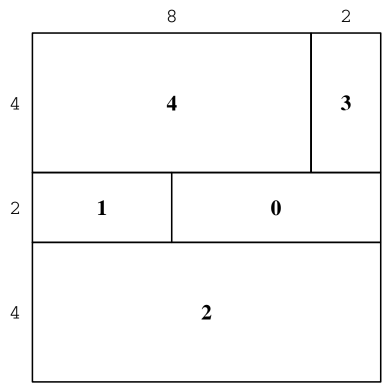


Figure 2: Top-Down Conversion of (D(D 2(R(0 1))(R 3 4)) for Dataset Simple-05

in Figure 1(c) will also be the same as the initial layout determined by Figure 1(a). Other researchers have argued that these redundancies lead to a richer search and better performance in an evolutionary GA problem solving framework [Kado, 1995]. We believe the same is true in an evolutionary GP framework. It has also been shown that the four operators are not redundant in other FLP variations that have obstructions within the site area [Kado, 1995].

There are two primary approaches for converting an STS-expression into a layout. Given a rectangular area equal to the total area required for all facilities, the "top-down" approach starts at the root node of the STS and recursively divides the available region into two subregions. The placement of a division, which is horizontal if the operator is U or B and vertical if the operator is L or R, is calculated by computing the area subtotals for the facilities on either side of the root node. The approach is broadly illustrated in Figure 2 using the elementary data set named Simple-05 from Table 1 with the expression from Figure 1(a). The top-down approach is useful for other FLP variations that have room limitations and obstructions [Tam, 1992a]. Without post-processing, the top-down strategy does not necessarily produce rectangles that respect the aspect ratio constraints.

| Facility Number | Area | Aspect Ratio | Width | Height | Orien-tation |
|---|---|---|---|---|---|
| 0 | 12 | 0.75 | 3 | 4 | Fixed |
| 1 | 8 | 0.50 | 2 | 4 | Fixed |
| 2 | 40 | 1.60 | 8 | 5 | Fixed |
| 3 | 8 | 2.00 | 4 | 2 | Fixed |
| 4 | 32 | 0.50 | 4 | 8 | Fixed |

Table 1: Simple-05 Data

A "bottom-up" approach for converting an STS expression into a layout begins by assembling any two sibling rectangles represented by terminal nodes in the manner specified by their parent operator. We align the centers of the two rectangles horizontally if the parent operator is L or R, and vertically if the parent operator is U or D. As we recursively ascend the STS expression, we use the centers of rectangles that bound groups of rectangles to determine alignment. This approach is illustrated in Figure 3, again using the data set Simple-05 from Table 1 with the expression from Figure 1(a). The resulting facility layout is not necessarily optimal. However, if each initial rectangle satisfies its required aspect ratio and if the assembly does not change the shape of any rectangle, the resulting facility layout will, of course, be in compliance with all of the aspect ratio constraints. The bottom-up approach is useful for our FLP variation since we assume that there are no room limitations [Cohoon *et al.*, 1991].

Our approach for converting an STS expression into a layout is similar to the approach used by Kado. We believe there are two reasons for obtaining superior results with our techniques. The first is due to an improved technique for converting the STS expression into a facility layout. The second reason is due to using a Genetic Programming framework that does not constrain

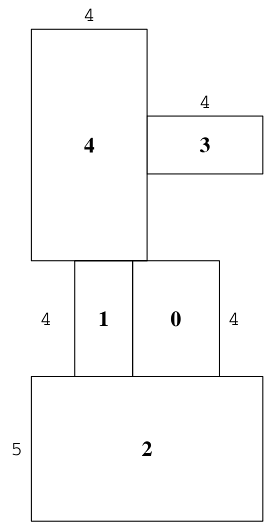


Figure 3: Bottom-Up Conversion of (D(D 2(R(0 1))(R 3 4)) for Dataset Simple-05

the structure of a facility layout. We refer to our approach as the Top-Down Bottom-Up (TDBU) conversion technique. The major steps of the TDBU technique for converting an STS expression to a facility layout are as follows:

1. Begin with a square region where the area of the square is equal to the total area required by all of the $n$ facilities. Partition the square into a collection of rectangles representing a temporary facility layout by using the STS and a top down approach.

2. Each rectangle resulting from Step 1 will have the correct area but will likely violate its aspect ratio constraint. We scale the entire square region resulting from Step 1 by $k * scaleW$ in the direction of the the width and $k * scaleH$ in the direction of the height. We choose $scaleW$ so that the width of each rectangle in the resulting scaled region is greater than or equal to the minimum width that is required for its corresponding facility, with either orientation, to be in compliance with its aspect ratio range. We select $scaleH$ in a similar manner. It is important to note that even if the new width of each scaled rectangle exceeds the product of its old width and $scaleW$ and even if the new height of each scaled rectangle exceeds the product of its old height and $scaleH$, it is not guaranteed that each scaled rectangle contains, as a subset, a subrectangle of the required area with aspect ratio in the required range. We choose k large enough so that the rectangles resulting from the next step, Step 3, have the required area, the required aspect ratio, and do not overlap. Two different values of k, as long as each is large enough, will determine the same facility layout.

3. Each rectangle resulting from Step 2 that is out of compliance with its aspect ratio range is contracted in one dimension as it is expanded in the other dimension to bring its aspect ratio into compliance, while leaving its center location and its area fixed. The direction for expansion and the amount of the expansion is determined in order to modify the rectangle as little as possible. Each rectangle, now in compliance with its required aspect ratio range, is then contracted in both dimensions in a way that exactly preserves the aspect ratio while reducing its area to the area required by the associated facility.

4. The non-overlapping rectangles resulting from Step 3 are packed to remove empty space using the STS and the bottom-up conversion technique.

## 4 Genetic Algorithms and Genetic Programming

Using the conventional genetic algorithm, individuals in a population are usually fixed-length character strings. Specification of the representation scheme starts with the selection of the string length and the alphabet size. The representation scheme maps each possible character string to a candidate solution in the search space of the problem. The evolutionary process starts with an initial population of randomly generated character strings and is driven by a fitness measure which assigns a fitness value to each possible character string. The GA works by selection, recombination, and mutation on the fixed length character strings [Koza, 1992].

The genetic program is an extension of the conventional genetic algorithm in which individuals in a population are expression trees (programs). Specification of the representation scheme starts with identifying a set of terminals and a set of functions. The representation scheme "evaluates" each possible expression tree to a candidate solution in the search space of the problem. The evolutionary process starts with an initial population of randomly generated expression trees and is driven by a fitness measure which assigns a fitness value to each possible expression tree. The GP works by selection, recombination, and mutation on expression trees [Kinnear, 1994, Koza, 1992, Koza, 1994].

## 5 Data Sets, Techniques, and Results

Eight different data sets with 5, 6, 7, 8, 12, 15, 20, and 30 facilities, respectively, were used to establish the validity of our procedure. The data sets are based on a test bank provided in Nugent et al. [Nugent *et al.*, 1969] as modified by Tam and Li [Tam and Li, 1991] to add aspect ratio constraints. Although the modified problems have aspect ratio constraints, they have no room limitations and no obstructions. Tam and Li's approach employs a divide and conquer strategy which first partitions the facilities into different clusters. The layout of each cluster is generated, and the overall layout is derived by treating each cluster as a large facility. Tam and Li's procedure uses an expensive sequential Lagragian method that is supplemented with a quasi-Newton procedure [Kado *et al.*, 1995]. Their procedure could not practically be used on the two largest problems in the test bank. The best flow obtained by Tam and Li for the other six problems is recorded in the TL91 column of Table 2.

Kado et al. [Kado *et al.*, 1995] provide a comprehensive performance evaluation of numerous variations of a two-phase optimization technique that uses cluster analysis and genetic algorithms. In each variation of their technique, clustering is used to fix the structure of a slicing tree, at least for the generation of an initial population of linear, fixed-length chromosomes that are used in their GA search. Their GA, with one or more of a variety of crossover and mutation operators is used to evolve their best reported flows. Their best flow for each of the eight test problems is indicated in the KA95 column of Table 2. Although partially due to a desire to narrow the search space, another motivation for using clustering techniques to create a fixed STS is to force an expression tree into a linear, fixed length chromosome so that standard crossover and mutation operators make sense. As an example of the problem, one can easily show that the uniform GA crossover applied to two postfix expressions, each corresponding to a different STS, does not necessarily produce valid postfix expressions. A variety of strategies are required to create and manipulate reasonable chromosomes representations using standard GA search strategies.

Our research also uses a Slicing Tree Structure to represent a candidate solution to the Facility Layout Problem. Our STSs are manipulated by a GP. The elegance of our technique is simply that it is more natural to use a GP that represents and manipulates STSs as trees than it is to use a GA that represents and manipulates STSs as linear chromosomes. The set of terminal nodes, corresponding to the set of facility labels, is the set {0, 1, ..., $n$-1}. The set of function nodes, corresponding to placement of the facilities (or bounding rectangles of collections of facilities) relative to each other, is the set {U, D, L, R}. Each expression trees is constrained to have exactly $n$ terminal nodes and each element of the set {0, 1, ..., $n$-1} must be represented in exactly one terminal node. The GP kernel used for our research is Gpc++, Version 0.40 [Fraser, 1994]. Our results are shown in the GP96 column of Table 2.

Our results are significantly superior to Kado's for the two largest 20 and 30 facility problems. Our results are equal or better that Kado's for four of the other six benchmark problems. After experimentation, each of our flow values listed in the GP96 column of Table 2 was obtained using the Gpc++ parameters indicated in Table 2. The *Pop* column indicates the size of a population in the genetic program. The *Gen* column indicates the number of generations that the genetic program was allowed to run. The *First* column indicates the generation when the reported result first occurred. The crossover rate is indicated as a percentage in the *Cr* column and the mutation rate is indicated as a percentage in the *Mu* column. If mutation was indicated, either a single internal (operator) node was changed, two subtrees were swapped, or two internal nodes were swapped, with probability indicated, respectively, by the percentages in column *M1*, *M2*, and *M3*. If crossover was indicated, a tournament selection was used to select parents. Since any valid STS has exactly $n$ terminal nodes, crossover points in the two parents were selected by repeated attempts, if required, to insure that the number of terminal nodes in each tree remains the same. Finally, since each element of the set {0, 1, ..., $n$-1} must occur exactly once in a valid STS tree, a deterministic procedure is applied to alter some terminal nodes in each tree resulting from crossover.

The data set for the TL91-05 benchmark problem is presented in Table 3. Our facility layout for TL91-05 is presented in Table 4. Table 4 gives the coordinates of the center, the width, the height, the area, and the two aspect ratios for each of the five facilities. Figure 4 graphically displays the facility layout for TL91-05. Our value of flow for TL91-05 is 226 (see Table 2). The data set for the TL91-20 and other benchmark problems is in the literature [Kado, 1995]. Our facility layout for TL91-20 is presented in Table 5. Figure 6 graphically displays the facility layout for TL91-20. Our value of flow for TL91-20 is 14,033 (see Table 2), vastly superior to Kado's result of 16,393. In fact, the results shown in Table 2 clearly show the superior results of our techniques in six (6) of the eight (8) benchmark problems, including the two larger data sets. Figure 5 shows the best and the average fitness values for each generation of our GP when applied to TL91-20. Our facility layout for TL91-30 is presented in Table 6. Figure 7 graphically displays the facility layout for TL91-30. Our value of flow for TL91-30 is 39,018 (see Table 2), also superior to Kado's result of 41,095. Corresponding to the GP96 flow values shown in Table 2, the generating slicing tree programs are shown in Table 7.

| Facility | Flow | | Aspect Ratio | | Orien- |
|---|---|---|---|---|---|
| Number | (Traffic) | Area | Low | High | tation |
| 0 | 0 5 2 4 1 | 24 | 0.80 | 1.00 | Free |
| 1 | 5 0 3 0 2 | 16 | 0.75 | 1.15 | Free |
| 2 | 2 3 0 0 0 | 36 | 0.60 | 1.85 | Free |
| 3 | 4 0 0 0 5 | 8 | 0.30 | 1.10 | Free |
| 4 | 1 2 0 5 0 | 21 | 0.90 | 1.18 | Free |

Table 3: TL91-05 Benchmark Data

| Problem | TL91 | KA95 | GPK | GP96 | Pop | Gen | First | Cr | Mu | M1 | M2 | M3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TL91-5 | 247 | 228 | 228 | 226 | 600 | 1,000 | 24 | 50 | 50 | 15 | 70 | 15 |
| TL91-6 | 514 | 361 | 361 | 384 | 600 | 1,000 | 33 | 50 | 50 | 15 | 70 | 15 |
| TL91-7 | 559 | 596 | 615 | 568 | 600 | 1,000 | 175 | 50 | 50 | 15 | 70 | 15 |
| TL91-7 | 559 | 596 | 596 | 595 | 1,500 | 4,000 | 979 | 50 | 50 | 15 | 70 | 15 |
| TL91-8 | 839 | 878 | 878 | 878 | 1,500 | 4,000 | 1,657 | 50 | 50 | 15 | 70 | 15 |
| TL91-12 | 3,162 | 3,283 | 3,576 | 3,220 | 1,500 | 3,000 | 168 | 70 | 40 | 25 | 50 | 25 |
| TL91-15 | 5,862 | 7,384 | 7,612 | 7,510 | 1,500 | 3,000 | 1,383 | 70 | 40 | 25 | 50 | 25 |
| TL91-20 | — | 16,393 | 14,026 | 14,033 | 1,500 | 3,000 | 2,521 | 50 | 50 | 15 | 70 | 15 |
| TL91-30 | — | 41,095 | 39,176 | 39,018 | 1,500 | 3,000 | 2,933 | 75 | 40 | 25 | 50 | 25 |

Table 2: Flows for TL91 Facility Flow Benchmark Problems

| Facility | XC | YC | Width | Height | Area | Ht/Wt | Wt/Ht |
|---|---|---|---|---|---|---|---|
| 0 | 5.8171 | 6.7685 | 4.7060 | 5.0998 | 24.0 | 1.0837 | 0.9228 |
| 1 | 1.7321 | 6.7685 | 3.4641 | 4.6188 | 16.0 | 1.3333 | 0.7500 |
| 2 | 4.0851 | 11.6134 | 7.8434 | 4.5899 | 36.0 | 0.5852 | 1.7088 |
| 3 | 6.5740 | 2.1093 | 2.1637 | 3.6974 | 8.0 | 1.7088 | 0.5852 |
| 4 | 3.0032 | 2.1093 | 4.9780 | 4.2186 | 21.0 | 0.8475 | 1.1800 |

Table 4: GP96 Facility Layout for the TL91-05 Benchmark Problem

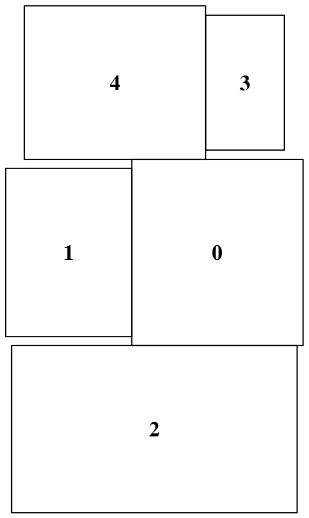


Figure 4: GP96 Facility Layout for the TL91-05 Benchmark Problem

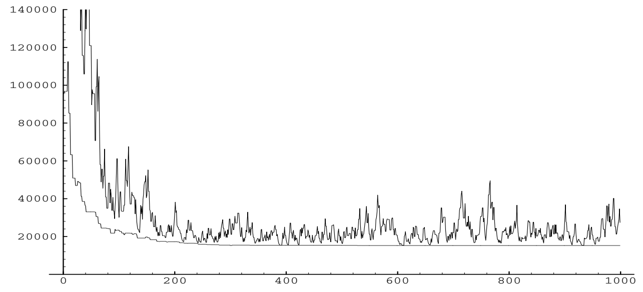


Figure 5: GP96 Number of Generations Versus Fitness Values for the TL91-20 Benchmark Problem

For each data set, the result reported in the GP96 column of Table2 was obtained by applying our variation of Kado's technique (for converting an STS expression to a facility layout as indicated in step 1 through step 4 above) to the best chromosome generated by our GP evolution. The corresponding result reported in the GPK column of Table 2 was obtained by applying Kado's STS conversion technique (using, in fact, Kado's own conversion code) to the best chromosome generated by our GP evolution. Examination of the GPK column and the GP96 column in the TL91-6 row and the first of the two TL91-7 rows in Table 2 indicate that the STS conversion technique used by Kado and the technique used by this research are not the same. The inclusion of the two TL91-7 rows in Table 2, corresponding to different GP parameter settings, is intentional since there is a significant contrast between the results in the GP96 and the GPK columns.

Although the STS trees generated by Kado's GA technique are not reported in our tables, we also note that Kado's evolved STS tree is identical to the STS tree generated by our GP technique for four of the rows in Table 2: TL91-5, TL91-6, the second TL91-7 row, and TL91-8.

| Facility | XC | YC | Width | Height | Area | Ht/Wt | Wt/Ht |
|---|---|---|---|---|---|---|---|
| 0 | 11.0826 | 14.2293 | 5.4443 | 4.4083 | 24.0 | 0.8097 | 1.2350 |
| 1 | 6.5279 | 2.4190 | 3.4641 | 4.6188 | 16.0 | 1.3333 | 0.7500 |
| 2 | 2.3979 | 10.8775 | 4.4113 | 8.1609 | 36.0 | 1.8500 | 0.5405 |
| 3 | 9.1856 | 2.6868 | 1.6803 | 4.7610 | 8.0 | 2.8334 | 0.3529 |
| 4 | 16.0265 | 9.8863 | 4.2186 | 4.9779 | 21.0 | 1.1800 | 0.8475 |
| 5 | 16.0265 | 14.4587 | 4.1998 | 4.1669 | 17.5 | 0.9922 | 1.0079 |
| 6 | 9.2599 | 8.9128 | 1.7990 | 2.0011 | 3.6 | 1.1123 | 0.8990 |
| 7 | 12.0826 | 9.9128 | 3.6453 | 4.2247 | 15.4 | 1.1589 | 0.8629 |
| 8 | 16.0265 | 2.3570 | 4.2426 | 4.7140 | 20.0 | 1.1111 | 0.9000 |
| 9 | 6.5279 | 13.5035 | 3.3288 | 5.8581 | 19.5 | 1.7598 | 0.5682 |
| 10 | 11.9700 | 2.2341 | 3.7639 | 4.2509 | 16.0 | 1.1294 | 0.8854 |
| 11 | 6.5279 | 9.1514 | 3.1623 | 2.8460 | 9.0 | 0.9000 | 1.1111 |
| 12 | 16.0266 | 6.0557 | 3.3541 | 2.6833 | 9.0 | 0.8000 | 1.2500 |
| 13 | 2.3979 | 4.1906 | 4.7958 | 5.2129 | 25.0 | 1.0870 | 0.9200 |
| 14 | 11.1484 | 5.2894 | 2.1508 | 1.8598 | 4.0 | 0.8647 | 1.1565 |
| 15 | 13.0454 | 5.2894 | 1.6432 | 1.8257 | 3.0 | 1.1111 | 0.9000 |
| 16 | 9.2599 | 10.9133 | 2.0000 | 2.0000 | 4.0 | 1.0000 | 1.0000 |
| 17 | 6.5279 | 6.2284 | 3.0000 | 3.0000 | 9.0 | 1.0000 | 1.0000 |
| 18 | 9.1856 | 6.3350 | 1.7748 | 2.5354 | 4.5 | 1.4286 | 0.7000 |
| 19 | 11.9700 | 7.0099 | 3.1623 | 1.5811 | 5.0 | 0.5000 | 2.0000 |

Table 5: GP96 Facility Layout for the TL91-20 Benchmark Problem

## 6 Future Work

We are currently articulating the precise distinction between Kado's STS conversion technique and the one used by our research. We are also developing alternate crossover and mutation techniques for our genetic programming evolution of STS trees. The objective of alternate crossover and mutation techniques is to duplicate or enhance other genetic algorithm strategies used by Kado. Further, we are evaluating our techniques with alternate flow functions including, in particular, a flow function where the distance is not squared. Finally, we are applying our techniques to other benchmark data sets that are in the literature. Some of the data sets and reported results use Manhattan distance and others have room limitations or obstructions in the site area.

## 7 Conclusions

This research finds optimal or near optimal solutions to the Facility Layout Problem by applying standard GP techniques to a Slicing Tree Structure that determines a facility layout. As noted before, the elegance of our technique is that it is quite natural to use genetic programming techniques to manipulate slicing trees. The superior nature of our results is due, in part, to our single phase global optimization strategy that does not constrain the tree structure in comparison to, say, a two phase optimization technique that first optimizes the structure of a slicing tree, and, then, optimizes the placement of facilities by using the fixed structure. All of our current research is in the context of eight FLPs where each facility has an associated aspect ratio but there are no room limitations and no obstructions in the site area. Clearly, the facility layout problem appears to be more conducive to a genetic programming strategy than a genetic algorithm strategy and this is indicated by our results.

## References

[Cohoon *et al.*, 1991] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. S. Richards. Distributed genetic algorithms for the floorplan design problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(4):483–492, April 1991.

[Fraser, 1994] Adam Fraser. Genetic programming in C++. Technical Report 040, Cybernetic Research Institute, 1994.

[Kado *et al.*, 1995] K. Kado, P. Ross, and D. Corne. A study of genetic algorithm hybrids for facility layout problems. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 499–505. Morgan Kaufmann, 1995.

[Kado, 1995] K. Kado. An investigation of genetic algorithms for facility layout problems. Master's thesis, University of Edinburgh, 1995.

| FACILITY | XC | YC | WIDTH | HEIGHT | AREA | HT/WT | WT/HT |
|---|---|---|---|---|---|---|---|
| 0 | 6.7144 | 3.4736 | 4.3818 | 5.4772 | 24.0 | 1.2500 | 0.8000 |
| 1 | 6.6301 | 17.4374 | 4.6188 | 3.4641 | 16.0 | 0.7500 | 1.3333 |
| 2 | 2.3178 | 5.3531 | 4.4113 | 8.1609 | 36.0 | 1.8500 | 0.5405 |
| 3 | 5.7157 | 14.3383 | 2.9261 | 2.7341 | 8.0 | 0.9344 | 1.0702 |
| 4 | 2.1213 | 18.7712 | 4.2186 | 4.9780 | 21.0 | 1.1800 | 0.8475 |
| 5 | 15.5903 | 14.0473 | 5.9160 | 2.9580 | 17.5 | 0.5000 | 2.0000 |
| 6 | 9.8926 | 8.7108 | 1.7392 | 2.0699 | 3.6 | 1.1901 | 0.8402 |
| 7 | 10.7142 | 2.4149 | 3.1886 | 4.8298 | 15.4 | 1.5147 | 0.6602 |
| 8 | 2.1213 | 13.9252 | 4.2426 | 4.7140 | 20.0 | 1.1111 | 0.9000 |
| 9 | 14.0511 | 7.7601 | 3.2914 | 5.9246 | 19.5 | 1.8000 | 0.5555 |
| 10 | 6.7144 | 8.0917 | 4.2564 | 3.7590 | 16.0 | 0.8831 | 1.1323 |
| 11 | 17.1198 | 6.7601 | 2.8460 | 3.1622 | 9.0 | 1.1111 | 0.9000 |
| 12 | 13.9608 | 17.1613 | 2.7523 | 3.2700 | 9.0 | 1.1881 | 0.8417 |
| 13 | 13.7829 | 21.1942 | 5.2128 | 4.7958 | 25.0 | 0.9200 | 1.0870 |
| 14 | 16.5587 | 11.6444 | 2.1693 | 1.8439 | 4.0 | 0.8500 | 1.1765 |
| 15 | 11.8107 | 14.0473 | 1.6432 | 1.8258 | 3.0 | 1.1111 | 0.9000 |
| 16 | 17.1198 | 9.3412 | 2.0000 | 2.0000 | 4.0 | 1.0000 | 1.0000 |
| 17 | 5.7427 | 11.4712 | 3.0000 | 3.0000 | 9.0 | 1.0000 | 1.0000 |
| 18 | 8.1301 | 11.4712 | 1.7748 | 2.5354 | 4.5 | 1.4286 | 0.7000 |
| 19 | 11.7941 | 17.1614 | 1.5811 | 3.1623 | 5.0 | 2.0001 | 0.5000 |
| 20 | 6.6301 | 21.0133 | 4.3386 | 3.6878 | 16.0 | 0.8500 | 1.1765 |
| 21 | 10.7141 | 6.2528 | 3.1623 | 2.8460 | 9.0 | 0.9000 | 1.1111 |
| 22 | 10.7142 | 11.1570 | 3.1886 | 2.8226 | 9.0 | 0.8852 | 1.1297 |
| 23 | 15.4741 | 2.3999 | 5.2128 | 4.7958 | 25.0 | 0.9200 | 1.0870 |
| 24 | 14.3894 | 11.6444 | 2.1693 | 1.8439 | 4.0 | 0.8500 | 1.1765 |
| 25 | 11.5838 | 8.7108 | 1.6432 | 1.8257 | 3.0 | 1.1111 | 0.9000 |
| 26 | 10.0175 | 14.0473 | 2.0000 | 2.0000 | 4.0 | 1.0000 | 1.0000 |
| 27 | 16.8370 | 17.1613 | 3.0000 | 3.0000 | 9.0 | 1.0000 | 1.0000 |
| 28 | 10.1162 | 17.1614 | 1.7748 | 2.5355 | 4.5 | 1.4286 | 0.7000 |
| 29 | 8.0931 | 14.3383 | 1.8288 | 2.7341 | 5.0 | 1.4950 | 0.6689 |

Table 6: GP96 Facility Layout for the TL91-30 Benchmark Problem

[Kinnear, 1994] Kenneth E. Kinnear, Jr., editor. *Advances in Genetic Programming.* MIT Press, Cambridge, MA, 1994.

[Koza, 1992] John R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[Koza, 1994] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, Cambridge Massachusetts, May 1994.

[Kusiak and Heragu, 1987] A. Kusiak and S. Heragu. The facility layout problem. *European Journal of Operational Research*, 29:229–251, 1987.

[Nugent *et al.*, 1969] C. E. Nugent, T. E. Vollmann, and J. Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16:150–173, 1969.

[Otten, 1982] R. H. J. M. Otten. Automatic floorplan design. In *Proceedings of the 19th ACM-IEEE Design Automation Conference*, pages 261–267, 1982.

[Smith and Tate, 1993] A. Smith and D. Tate. Genetic optimization using a penalty function. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 499–505. Morgan Kaufmann, 1993.

[Tam and Li, 1991] K. Y. Tam and S. G. Li. A hierarchial approach to the facility layout problem. *International Journal of Production Research*, 29(1):165–184, 1991.

[Tam, 1992a] K. Y. Tam. Genetic algorithms, function optimization, and facility layout design. *European Journal of Operational Research*, 63(2):322–346, December 1992.

[Tam, 1992b] K. Y. Tam. A simulated annealing algorithm for allocating space to manufacturing cells. *International Journal of Production Research*, 30(1):63–87, 1992.

| Problem | GP96 Generated Program |
|---|---|
| TL91-5 | ((U(U(L 4 3)(L 1 0)) 2)) |
| TL91-6 | ((U(U(L 5(U 4 3))(L 1 0)) 2)) |
| TL91-7 | ((U(U 2(L 1(L 6 5)))(L(L 0 3) 4))) |
| TL91-7 | ((U 2(L(L(U 1 0)(U 6 3))(U 5 4)))) |
| TL91-8 | ((U(U 2(L(L 1 7) 0))(U(L 6 3)(L 5 4)))) |
| TL91-12 | ((U(U(U 4(L 5 9))(L(L 10(L 3(L 6 7))) 1))(U(L 11(L 8 0)) 2))) |
| TL91-15 | ((L(L(U 8 0)(U(L(U(U 10(L 7 6))(L 11 12))(U 4 13))(U(L(L 1 3) 14)2)))(U 5 9))) |
| TL91-20 | ((L(U 13 2)(L(U 1(U 17(U 11 9)))(L(U(L(U 3 18)(U 10(U(L 14 15) 19))) (U(L(U 6 16) 7) 0))(U(U(U 8 12) 4) 5))))) |
| TL91-30 | ((L(U(L 2(U 0 10))(L(U 8 4)(U(L 17 18)(U(L 3 29)(U 1 20))))) (U(U(U(L(U(U 7 21)(U(L 6 25) 22))(U 23(U(L 9(U 11 16))(L 24 14)))) (L(R 15 26) 5))(L(L 28 19)(L 12 27)))13))) |

Table 7: GP96 Generated Facility Layout Programs for TL91 Benchmark Problems

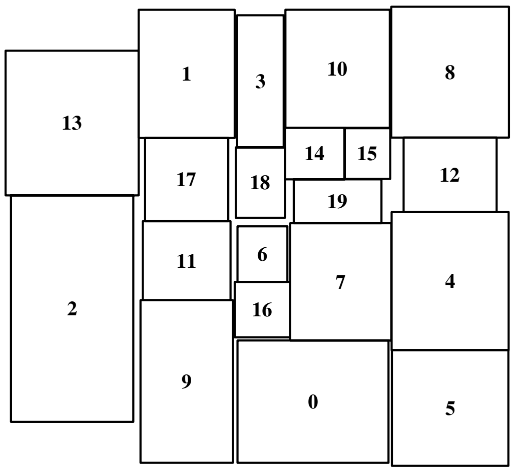


Figure 6: GP96 Facility Layout for the TL91-20 Benchmark Problem

[Wong and Liu, 1986] D. F. Wong and C. L. Liu. A new algorithm for floorplan design. In *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pages 101–107, 1986.

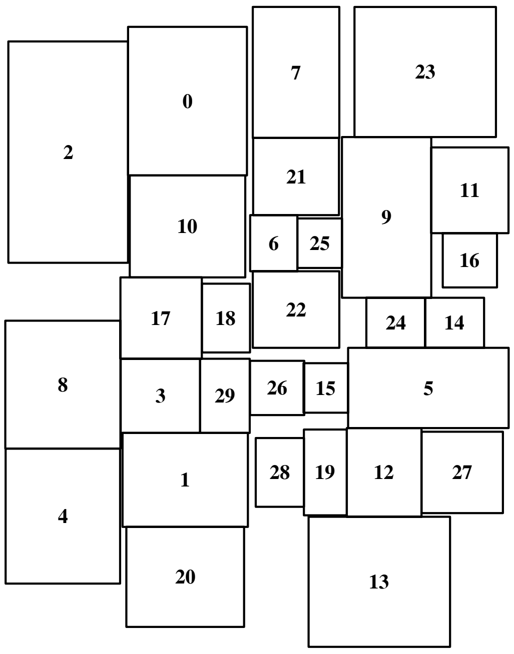


Figure 7: GP96 Facility Layout for the TL91-30 Benchmark Problem