

# CRITICAL NET ROUTING

J.P. Cohoon and L. J. Randall  
Department of Computer Science  
University of Virginia  
Charlottesville, VA

## ABSTRACT

A critical net has been routed traditionally by interconnecting its terminals with a minimum length rectilinear Steiner tree (MRST). We propose a new interconnection form, the maximum performance tree (MPT), that better approximates an interconnection with optimal circuit performance. In addition, we both present a heuristic approach that quickly generates MPTs and compare the quality of these trees with MRSTs for several net instances.

## INTRODUCTION

The most ubiquitous of routing problems is to determine an optimal interconnection for the terminals of a critical net. The traditional approach is to construct a minimum length rectilinear Steiner tree (MRST) [1, 4]. In constructing a MRST, it is permissible to introduce new points, called *Steiner points*, if their inclusion allows a tree with less length. Contrary to common belief, the Steiner approach for interconnecting the terminals of a critical net is not necessarily the best one. The approach can fail because its solution is optimal with respect to total wire length which is the wrong figure of merit — the true figure of merit is circuit performance. For critical nets, circuit performance is measured by the time it takes to drive the net's signal. Since a typical critical net has a single *source* terminal that outputs its signal to the remaining *sink* terminals, circuit performance can be better approximated by the length of the longest source-terminal-to-sink-terminal path than by total wire length.

It is relatively simple to construct an interconnection tree that optimizes the source-to-sink value by using shortest path algorithms to generate a shortest path tree [3]. However, it is far more difficult to choose from among such interconnection trees the one with minimum total length. We call such trees, *maximum performance trees* (MPT). Figure 1 depicts a MPT for a given net. The MPT has a longest source-to-sink value of 350 units and a total wire length of 1300 units (the source terminal is denoted by the white-filled square, other terminals are denoted by black-filled squares, Steiner points are denoted by bullets). The rectilinear minimum spanning tree

(RMST) for this instance has a length of 1295 units and a longest source-to-sink value of 700 units. In inducing heuristically a MRST from the RMST in the traditional manner [2, 6], we were unable to reduce the total amount of wire. In addition, the best found MRST with respect to the longest source-to-sink value required 450 units. Thus, total wire length is comparable for all three tree forms, but the MPT form has a source-to-sink value that is 50% smaller than that of the RMST form and approximately 25% smaller than that of the traditional Steiner tree form!

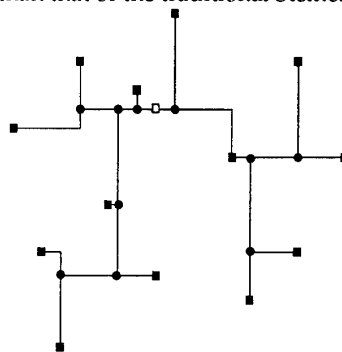


Figure 1. — Maximum Performance Tree

In the remainder of this paper, we first present a heuristic approach, *MP*, to generating maximum performance trees. In an experimental results section, we then compare the quality of *MP*'s maximum performance trees to those produced by the traditional Steiner approach. Finally, we discuss some possible improvements to our circuit performance net model.

## BASIC ALGORITHM

Our approach to generating MPTs has three phases. The approach is quite fast and can solve instances with several hundred terminals in a fraction of a second. Although the first phase is optional, it is generally recommended. The phase constructs a *trunk* for the interconnection tree. The remaining wire segments that complete the interconnection will be branches or sub-branches off the trunk. A *basis* for these remaining segments is constructed in the second phase and is improved upon in the third phase.

**Initial Phase: Trunk Generation** We have observed in our study of MPTs that the source-to-sink paths for the most distant sink terminals often form trunks in the optimal solution. Therefore, we have created several tools that generate trunks with specific properties. One common property to all the tools is that they generate a shortest path connection from the source to a most distant sink. Thus, the length of this interconnection is optimum with respect to the source-to-sink value.

Trunks can currently be generated in five different ways. Let  $s$  be the source terminal and  $t$  be the most distant sink. The first method averages the  $x$ -coordinates of all terminals that lie between  $s$  and  $t$  with respect to the  $x$ -axis. This average ( $x_{mean}$ ) is then used to generate the central vertical trunk segment that lies on the line  $x = x_{mean}$ . Two horizontal segments complete the trunk by connecting the central trunk segment to  $s$  and  $t$ . This is depicted in Figure 2. The second method is similar to the first, but

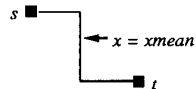


Figure 2. — Central trunk derived from  $x_{mean}$ .

replaces  $x_{mean}$  with the median of the  $x$ -coordinates of all terminals that lie between  $s$  and  $t$  ( $x_{median}$ ). The third and fourth methods produce respectively horizontal central trunk segments that lie on the line  $y = y_{mean}$  or the line  $y = y_{median}$ , where  $y_{mean}$  and  $y_{median}$  are defined, respectively, in a manner analogous to  $x_{mean}$  and  $x_{median}$ . The remainder of the trunk consists of vertical connection segments that link the central horizontal segment to  $s$  and  $t$ .

The fifth and final trunk generation method assumes a complete graph exists on the given set of terminals. A rectilinear shortest path tree (RSPT) algorithm is then run on this graph. From the RSPT, our algorithm includes as part of the trunk system, the path from the source to the most distant sink for each of the four rectilinear directions. This method is depicted below in Figure 3. Note that for this instance, the trunk system branches out in only three directions as the RSPT in question did not have a left-emaneating edge from source  $s$ .

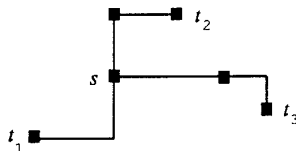


Figure 3. — Central trunk derived from the shortest path tree

Intuition and experience indicate that the first four methods produce their best solutions with

respect to total wire length when the source terminal lies on or near the net's bounding box. The fifth method's trunks are generally best for those instances where the source is located near the center of the net's bounding box.

**Middle Phase: Net completion** Once the trunk is established, the remainder of the net is realized using one of three techniques: a RMST algorithm, a RSPT algorithm, or a hybrid algorithm derived from a combination of the RMST and RSPT algorithms.

The RMST method is one that is commonly used to generate initial solutions to many routing problems since it generally produces interconnections that are good approximations to MRST. Experimental evidence indicates that RMST solutions are on average within 12% of the optimal MRST [6]. The major drawback of the RMST method is that it may produce a solution whose source-to-sink value is far from optimal. This was the case for the instance depicted in Figure 1, where the source-to-sink value is 50% less in a MPT than in a MRST derived from an RMST.

The basis of the MPT in Figure 1 was produced by generating an  $x_{mean}$ -derived trunk and completing the connection through a RMST. The basis is given in Figure 4 with the trunk being highlighted in bold. Although the basis tree and the MPT are similar, the MPT in question has had its trunk and several of its branches modified and re-attached. These modifications occurred in the final phase.

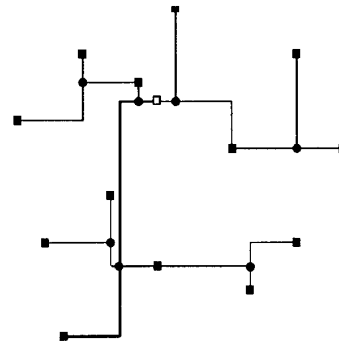


Figure 4. — RMST with  $x_{mean}$ -based trunk.

The second net completion method, using RSPTs, seems especially suited for the critical net problem since it produces solutions that minimize the source-to-sink value. However, its solutions have total wire length that is generally greater than that produced with the RMST.

The hybrid net completion algorithm is an attempt to combine the best features of an RMST and an RSPT. The method works in the following manner. First, an optimum source-to-sink value path  $l$  is com-

puted for the most distant sink terminal. Terminals are then successively added to the tree according to the rules of the RMST-based algorithm unless the resultant source-to-sink path to the terminal would produce a source-to-sink value greater than  $l$ 's source-to-sink value. In that case, the terminal is connected to the tree according to the RSPT rules.

Once the connection form has been determined by one of the above methods, the actual edges are then iteratively added to the tree in a greedy fashion. Since any rectilinear edge that contains both a horizontal and a vertical component (i.e., a *corner edge*) can be routed in two ways, both possible orientations are considered for each connection.

At each step, the edge is added that has the maximum possible overlap with the existing tree. In the case of a tie, the potential overlap of each contending edge is estimated with respect to the remaining un-added edges, and a contending edge with maximum potential is chosen. In the future, we plan to implement Ho, Vijayan, and Wong's algorithm that computes a minimum overlap tree given a set of interconnections [2].

**Final Phase: Tree Improvement** The final phase examines the net's complete interconnection in an attempt to reduce its total wire length. The reduction is achieved by applying a series of *corner flips* and edge insertions and deletions. These operations are performed in a matter that does not increase the interconnection's source-to-sink value for the most distant terminal. The phase terminates once the solution is locally optimal with respect to corner flips and edge insertions and deletions.

A *corner connection* is a right-angle connection such that the Steiner intersection point of its vertical and horizontal segments does not intersect any other segments (corner connections resemble corner edges). Therefore, every corner connection also has two valid orientations. Upon flipping a corner, one or both of the new corner segments may overlap previously placed segments, thus decreasing the total wire length by the amount of the overlap. Observe that a corner flip can never increase the total wire length or the longest-source-to-sink value. Improvement by edge deletion and insertion consists of first selecting an edge  $e$  from the current tree. We then consider the two disjoint sub-trees  $T_1$  and  $T_2$  that would be produced if edge  $e$  was deleted. Each node  $s$  in  $T_1$  ( $T_2$ ) is inspected and its distance to each edge in  $T_2$  ( $T_1$ ) is determined. The minimum such distance,  $d_s$ , for  $s$  is maintained. We then compute the shortest possible interconnection that can rejoin  $T_1$  and  $T_2$  by finding  $\min\{d_s\}$  over all  $s$  in  $T_1$  and  $T_2$ . If that interconnection neither increases the total wire

length nor the maximum source-to-sink value, the tree transformation is made permanent. By using union-find algorithms [3], the sub-tree examination process is quite efficient.

An edge insertion and deletion example is given in Figure 5 where edge  $e$  is deleted from the tree and edge  $f$  is inserted. The edge deletion and insertion

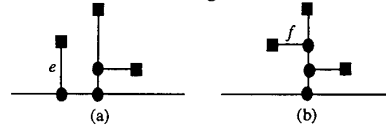


Figure 5. — Edge insertion and deletion.

operation permits an edge to be replaced by an edge of equal length if the maximum source-to-sink value is reduced. The insertion and deletion operations may also introduce corner connections. We observed that both events occurred frequently during our experiments.

Since the three-phased approach is able to construct a MPT in a fraction of a second for any reasonable-sized problem instance, we suggest that in practice the algorithm be run once for each of the fifteen possible trunk generation, net completion, and tree improvement combinations. The combination algorithm, called *MP*, would then return the best solution of the fifteen. In the experiments described below, we report as our results the solution determined by this combination method.

## EXPERIMENTS

Using the complete *MP* method on the critical net instance of Figure 1, we are able to produce a solution that matches the traditional RMST and Steiner solutions in terms of total wire length (1295 units), while maintaining our dramatic improvement with respect to the source-to-sink value for the most distant terminal. The overall best MPT was constructed using a *ymedian*-derived central trunk and the RSPT net completion method. When we omitted the initial phase (i.e., no pre-constructed central trunk for the net completion phase), then the best maximum performance tree we could construct had a total wire length of 1320 units. This tree was generated using the hybrid net completion heuristic.

We also ran *MP* on a variety of other input instances and its performance there was quite comparable. On average, *MP* produces solutions that are 25% better than Steiner trees with respect to the source-to-sink metric. This marked improvement requires on average only 6% more total wire than that used in the Steiner tree method.

The quality of *MP*'s solutions is also demonstrated in Figures 6 and 7. Figure 6 shows the basis

tree for the MPT given in Figure 7. The MPT in question uses 1424 units of wire in total and has a source-to-sink value of 475 units for the most distant terminal. The basis was constructed using a *y*-mean-derived central trunk and the RMST net completion method. The MPT compares very favorably to the best found MRST for this instance which has a longest source-to-sink value of 650 units and 1350 units of wire in total.

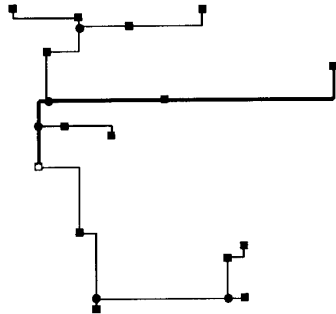


Figure 6. — A basis tree.

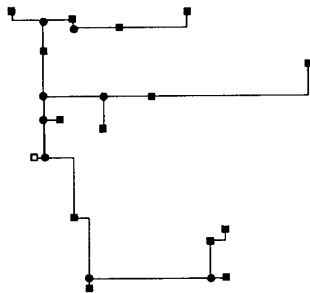


Figure 7. — A maximum performance tree for the basis tree of the previous figure.

### CURRENT RESEARCH

We are investigating various other heuristics for all three phases of the algorithm. We believe that this research will result in an algorithm that not only optimizes the source-to-sink metric, but also better minimizes the difference in total wire length between a MPT solution and a traditional MRST solution.

We are also investigating improvements to our circuit performance model. For example, we are currently pursuing a variation that does not minimize source-to-sink length but instead minimizes the longest source-to-sink delay. This delay can be satisfactorily modeled by assigning a weight to each terminal that is proportional to the delay associated with its circuit element. The cost of a wire segment is not measured in distance, but in the time it takes to traverse the segment. The cost of a path is then the sum of the edge costs and the weights of the path's internal vertices. The goal is to find a minimum-cost interconnection among those trees that minimize the maximum cost path. A heuristic that successfully

found such a path would also be useful as a routing estimator in performance-driven placement [5].

### SUMMARY

We have proposed a new critical net interconnection form, the *maximum performance tree*, that better approximates an interconnection with optimal circuit performance than the traditional minimum length Steiner tree model. In addition, we have presented a heuristic approach that quickly generates maximum performance trees. Our experimental results show that our heuristic approach produces trees that are comparable to Steiner trees with respect to total wire length, but can be significantly better with respect to signal driving time.

### ACKNOWLEDGEMENTS

This research was supported in part by the Virginia Center for Innovative Technology through grant 5-30971. Their support is greatly appreciated.

### REFERENCES

- [1] M. R. Garey and D. S. Johnson, The Rectilinear Steiner Tree Problem is NP-Complete, *SIAM Journal of Applied Mathematics* vol. 32, 1977.
- [2] J.-M. Ho, G. Vijayan, and C. K. Wong, New Algorithms for the Rectilinear Steiner Tree Problem, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. cad-9(2), pp. 185-193, February 1990.
- [3] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Potomac, MD, 1978.
- [4] F. K. Hwang, On Steiner Minimal Trees with Rectilinear Distance, *SIAM Journal of Applied Mathematics*, vol. 30, pp. 104-114, 1976.
- [5] M. Marek-Sadowska and S. P. Lin, Timing Driven Placement, *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, pp. 94-96, 1989.
- [6] D. S. Richards, Fast Algorithms for Rectilinear Steiner Trees, *Algorithmica*, vol. 4, pp. 191-207, 1989.