Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

PROVABLY GOOD PERFORMANCE-DRIVEN GLOBAL ROUTING

Jason Cong
Andrew B. Kahng
Gabriel Robins

April 1991
CSD-910013

Provably Good Performance-Driven Global Routing

Jason Cong, Andrew Kahng, and Gabriel Robins

Dept. of Computer Science, UCLA
Los Angeles, CA 90024

"There is timing in everything."
– Miyamoto Musashi, 1584-1645
A Book of Five Rings

Abstract

In this paper, we propose a provably good performance-driven global routing algorithm for both cell-based and building block design. The algorithm simultaneously minimizes both routing cost and the longest interconnection path, so that both are bounded by small *constant* factors away from optimal. Our method is based on the following two results. First, for any given value of a parameter ϵ, we can construct a routing tree with cost at most $(1+\frac{1}{\epsilon})$ times the minimum spanning tree weight, and with longest interconnection path length at most $(1+\epsilon)\cdot R$. Moreover, for Steiner global routing in arbitrary weighted graphs, we achieve wiring cost within a factor $2\cdot(1+\frac{1}{\epsilon})$ of the *optimal Steiner tree* cost, again with longest path length at most $(1+\epsilon)\cdot R$. In both cases, R is the minimum possible length from the source to the furthest sink. We also show that geometry helps in routing: in the Manhattan plane, the total wirelength for Steiner routing improves to $\frac{3}{2}\cdot(1+\frac{1}{\epsilon})$ times the optimal Steiner tree cost, while in the Euclidean plane, the total cost is further reduced to $\frac{2}{\sqrt{3}}\cdot(1+\frac{1}{\epsilon})$ times optimal. Extensive simulations confirm that this approach works well, using a large set of examples which reflect both cell-based and building-block layout styles.

1 Introduction

With progress in VLSI fabrication technology, interconnection delay has become increasingly significant in determining circuit speed. Recently, it has been reported that interconnection delay contributes up to 50% to 70% of the clock cycle in the design of dense, high performance circuits [5] [23]. Thus, with submicron device dimensions and up to a million transistors integrated on a single microprocessor, on-chip and chip-to-chip interconnections play a major role in determining the performance of digital systems.

Due to this trend, performance-driven layout design has received increased attention in the past several years. Most of the work in this area has been on the timing-driven placement problem, where a number of methods have been developed for placing blocks or cells in timing-critical paths close

together. The so-called zero-slack algorithm was proposed by Hauge, Nair and Yoffa [11]; fictitious facilities and floating anchors methods were used by Marek-Sadowska and Lin [18], and a linear programming approach was used by Jackson, Srinivasan and Kuh [13] [14]. Several other approaches, including simulated annealing, have also been studied [5] [17] [23]. Since no global routing solution is generally available at the placement step, most of these placement algorithms use the net bounding box semiperimeter to estimate the interconnection delay of a net.

While such techniques have been developed for timing-driven placement, only limited progress has been reported for the timing-driven interconnection problem. In [6], net priorities are determined based on static timing analysis; nets with high priorities are processed earlier using fewer feedthroughs. In [15], a hierarchical approach to timing-driven routing was outlined. In [19], a timing-driven global router based on the A* heuristic search algorithm was proposed for building-block design. However, these results do not provide a general formulation of the timing-driven global routing problem. Moreover, their solutions are not flexible enough to provide a *trade-off* between interconnection delay and routing cost.

Recently, we have proposed [2] a new model of timing-driven global routing for cell-based design, based on the idea of finding minimum spanning trees with bounded radius. The method in [2] constructs a spanning tree with radius $(1+\epsilon)\cdot R$ by using an analog of the classical Prim minimum spanning tree (MST) construction, where R is the minimum possible tree radius and ϵ is a user-specified parameter. Such an approach offers a very natural, smooth trade-off between the tree radius (maximum signal delay) and the tree cost (total interconnection length). This gives the circuit designer a great deal of algorithmic flexibility, as the parameter ϵ can be varied depending on performance constraints. Empirical performance results were very good, e.g., an average of 25% reduction in longest source-sink path was reported for 10-pin nets. However, the total wire cost using this method can be an unbounded factor worse than optimal.

In this paper, we propose a new method for timing-driven global routing. Our global router is based on a *provably good* algorithm that *simultaneously* minimizes total wirelength and maximum delay. More specifically, given a positive real parameter ϵ and a set of terminals, our method produces a routing tree with total cost at most $\frac{1}{\epsilon}\cdot$ times the MST cost, and with radius at most $(1+\epsilon)\cdot R$. In other words, both the total wirelength and the maximum delay of the routing are simultaneously bounded by small

constant factors away from their optimal values. The method applies to building-block layout styles in addition to the more geometric cell-based designs. In fact, our method generalizes to arbitrary weighted graphs, and also to Steiner routing formulations, where we achieve a wirelength bound of $2 \cdot (1 + \frac{1}{\epsilon})$ times the optimal *Steiner* tree cost, while still observing the $(1 + \epsilon) \cdot R$ radius limit.

We then show that geometry helps in routing: in the Manhattan plane, our wirelength bound for Steiner routing can be improved to $\frac{3}{2} \cdot (1 + \frac{1}{\epsilon})$ times optimal, and in the Euclidean plane, the Steiner routing bound improves further to $\frac{2}{\sqrt{3}} \cdot (1 + \frac{1}{\epsilon})$ times optimal. This series of results is especially surprising since constructing a minimum spanning tree with bounded radius in a general graphs is NP-complete [12], as is the Steiner problem in graphs [10].

Our construction can minimize either total wirelength (a minimum spanning tree) or the longest source-sink path (a minimum delay or minimum radius tree), depending respectively on whether we set $\epsilon = \infty$ or $\epsilon = 0$. Between these two extremes, the method offers a continuous, smooth tradeoff. In practice, our algorithm exhibits very good empirical performance which confirms this smooth tradeoff between the competing requirements of minimum delay and minimum total wirelength.

The remainder of this paper is organized as follows. In Section 2, we present the general formulation of the performance-driven global routing problem, and outline the very natural heuristic construction of [2]. In Section 3, we present a new and effective heuristic algorithm for computing bounded radius routing trees, and show that the algorithm is provably good with constant-factor performance bounds with respect to both delay and routing cost. Section 4 generalizes our method to Steiner tree global routing, and experimental results are reported in Section 5.

2 The Problem Formulation

A signal net N is a set of terminals, with one terminal s a designated source and the remaining terminals sinks. Because terminals of a signal net can be embedded in the Manhattan plane (for standard-cell or sea-of-gates design) or within a channel intersection graph (for macro-cell or block design), the global routing problem can have two distinct flavors. In the former case, the cost of routing between two nodes is given by geometric distance, while in the second case the cost is the total edge cost of the

shortest path between the nodes.[1] With this in mind, we define the underlying *routing graph* to be a connected weighted graph $G = (V, E)$. Each net is a subset of the nodes in this graph. A routing solution of a net is a tree in G, which we call the *routing tree* of the net, connecting all the nodes in N.

Since the routing tree may be treated as a distributed RC tree, we may use the first-order moment of the impulse response (also called Elmore's delay) to approximate interconnection delay [8] [22]. A more accurate approximation can be obtained using the upper and lower bounds on delay in an RC tree derived in [22]. However, although both the formula for Elmore's delay and those in [22] are very useful for simulation or timing verification, they involve sums of quadratic terms and are difficult to compute and optimize during the layout design process.

Thus, a linear RC model (where interconnection delay between a source and a sink is proportional to the wirelength between the two terminals) is often used to derive a simpler approximation for interconnection delay (e.g., [17] [21]). In this paper, we shall also use wirelength to approximate interconnection delay in the construction of routing solutions. In practice, a subsequent iterative improvement step, based on a more accurate RC delay model, may be used to enhance the routing solutions.

We say that the *cost* of a path in G is the sum of the edge weights in the path. The *shortest path* in G between two terminals $x, y \in N$, denoted by $minpath_G(x, y)$, is the path connecting x and y with smallest cost. In a routing tree T, $minpath_T(x, y)$ is simply the unique path between x and y. Note that in the geometric case, the cost of $minpath_G(x, y)$ is simply geometric distance, and we use the notation $dist(x, y)$ for clarity. For a weighted graph G we use $dist_G(x, y)$ to denote the cost of $(minpath_G(x, y))$.

Definition: The *radius* R of a signal net is the cost of a shortest path in G from the source to the furthest sink, i.e., $max_{x \in N}\ dist_G(s, x)$.

Definition: The *radius* of a routing tree T, denoted by $r(T)$, is the cost of a (shortest) path in T from the source s to the furthest sink. Clearly, $r(T) \geq R$ for any routing tree T.

According to the linear RC delay model, we minimize the interconnection delay of a net by min-

[1] For simplicity of presentation, our definition of the cost of an edge reflects only the wirelength. It is straightforward to extend the cost definition to be a function of wirelength, channel capacity and current channel density. The results presented in the next two sections will still hold.

imizing the radius of the routing tree, which measures the maximum interconnection delay between the source and any sink. On the other hand, we also want a routing tree with small total wirelength. Without this latter consideration, we could simply use the ***shortest path tree*** (SPT) of the net, i.e., the union of all the shortest source-sink paths computed by Dijkstra's single-source shortest path algorithm [20]. Although the radius $r(SPT)$ has the smallest possible radius of any routing tree, the SPT cost might be very high: Figure 1 shows a case where the cost of the shortest path tree can be $\Omega(|N|)$ times greater than the cost of the minimum spanning tree.

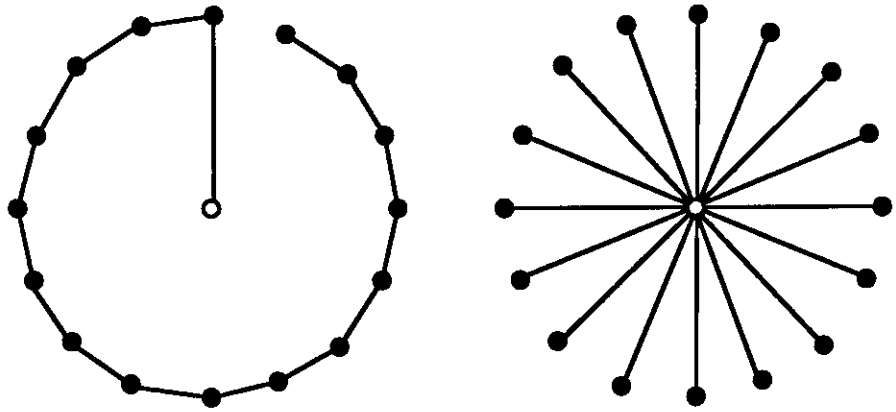


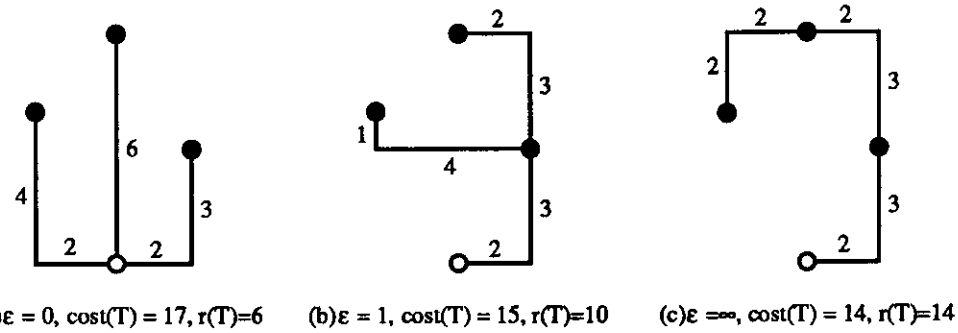
Figure 1: An example where the cost of a routing tree (right) is $\Omega(|N|)$ times larger than the cost of a minimum spanning tree (left).

A routing tree with high cost may increase the overall routing area. Moreover, high cost also contributes to the interconnection delay which is not captured in the linear RC model. Therefore, neither tree shown in Figure 1 is particularly desirable. In order to balance the radius and the cost in the routing tree construction, we formulate the timing-driven global routing problem as follows:

The Bounded Radius Minimum Routing Tree (BRMRT) Problem: Given a parameter $\epsilon \geq 0$ and a signal net with radius R, find a minimum-cost routing tree T with radius $r(T) \leq (1+\epsilon) \cdot R$.

The parameter ϵ controls the trade-off between the radius and the cost of the tree. When $\epsilon = 0$, we minimize the radius of the routing tree and thus obtain a shortest path tree for the signal net; on the other hand, when $\epsilon = \infty$ we minimize the total cost of the tree and obtain a minimum spanning tree. In general, as ϵ grows, there is less restriction on the radius, allowing further reduction in tree cost. Figure 2 shows an example where three distinct spanning trees are obtained using different values of ϵ: Figure 2(a) shows the minimum radius spanning tree corresponding to the case $\epsilon = 0$, with maximum

pathlength $r(T) = 6$; Figure 2(b) shows a solution with $r(T) = 10$ corresponding to the case $\epsilon = 1$; and Figure 2(c) shows the minimum cost spanning tree corresponding to the case $\epsilon = \infty$, with $r(T) = 14$.



(a)ε = 0, cost(T) = 17, r(T)=6 (b)ε = 1, cost(T) = 15, r(T)=10 (c)ε =∞, cost(T) = 14, r(T)=14

Figure 2: An example of how increasing the value of ϵ may result in decreased tree cost, but increased radius $r(T)$.

The bounded-radius minimum spanning tree formulation provides a great deal of flexibility for our timing-driven global router. In practice, the router uses small ϵ for nets in timing-critical paths, minimizing interconnection delay. For nets not in any timing-critical path, the router may use large ϵ so that the total wirelength is minimized.

We now briefly summarize the result in [2], which applies to cell-based layout and thus assumes that $N = V$, and that G is simply the complete graph over the set N of terminals, with edge weight equal to geometric distance.

The basic algorithm finds a routing tree by growing a single component, following the general scheme of Prim's classical minimum spanning tree construction. We grow a tree T which initially contains only the source s. At each step, we choose $x \in T$ and $y \notin T$ such that $dist(x, y)$ is minimum. If adding the edge (x, y) to T does not violate the radius constraint, i.e., $dist_T(s, x) + dist(x, y) \leq (1 + \epsilon) \cdot R$, we include the edge (x, y) in T. Otherwise, we *backtrace* along the path from x to s in T to find the first point x' such that (x', y) is *appropriate* (i.e., $dist_T(s, x') + dist(x', y) \leq R$), and add (x', y) to the tree.

In the worst case, the backtracing will terminate with $x' = s$, since the edge (s, y) is always appropriate. Note that in backtracing we could have chosen x' using the weaker criterion $dist_T(s, x') + dist(x', y) \leq (1 + \epsilon) \cdot R$. However, our choice of appropriate edges leads to fewer backtracing operations, while guaranteeing that backtracing is still always possible. In other words, we intentionally introduce some "slack" at y so that points within an ϵR neighborhood of y will not cause additional backtracing.

Limiting the amount of backtracing in this way will keep the cost of the resulting tree close to that of the minimum spanning tree. This algorithm is called the **Bounded Prim** (BPRIM) construction in [2].

Empirical results of the BPRIM method are very promising, as reported in [2]. However, it is somewhat unsatisfactory that the BPRIM method has unbounded worst-case performance ratio: for any value of ϵ, it can yield a routing tree with radius less than $(1+\epsilon) \cdot R$, but with cost $\Theta(|N|)$ times optimal [2]. Thus, the next section develops a new, provably good approach to performance-driven global routing.

3 Bounded Radius Spanning Tree Global Routing

This section describes a new algorithm which finds a provably good routing solution based on a spanning tree construction. We may consider the interterminal distances to be given by $G = (V, E)$ with $V = N$. This is most appropriate when distances between nodes are given by geometric distance, as occurs in global routing for cell-based design. For this case, many global routing methods are based on constructing a spanning tree for each net (e.g., see [3]). Therefore, the BRMRT problem becomes the **Bounded Radius Minimum Spanning Tree** (BRMST) problem.

The basic idea of our bounded radius minimum spanning tree algorithm is to construct a subgraph Q which spans G and has both small total cost and small radius. Thus, the shortest path tree of Q will also have small cost and radius, and corresponds to a good routing solution. Our algorithm is as follows.

- Let SPT_G be the shortest path tree of G, let MST_G be the minimum spanning tree of G, and initialize the graph Q to be equal to MST_G.
- Let L be the sequence of vertices corresponding to a depth-first tour of MST_G, where each edge of MST_G is traversed exactly twice (see Figure 3). The total edge cost of this tour is twice that of MST_G.
- Traverse L while keeping a running total S of traversed edge costs. As this traversal reaches each node L_i, check whether S is greater than $2\epsilon \cdot dist_{SPT_G}(s, L_i)$. If so, reset S to 0 and merge

$minpath_{SPT_G}(s, L_i)$ into Q. Continue traversing L while repeating this process.

- Our final routing tree is SPT_Q, the shortest path tree over Q.

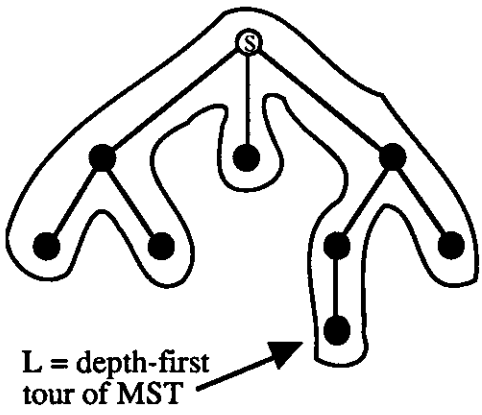


Figure 3: A spanning tree and a depth-first tour.

The formal description of the algorithm is given in Figure 4.

```
E' = edges of MST_G
Q = (V, E')
L = depth-first tour of MST_G
S = 0
for i = 1 to |L| - 1
    S = S + cost(L_i, L_{i+1})
    if S ≥ 2ε · dist_G(s, L_{i+1}) then
        E' = E' ∪ minpath_G(s, L_{i+1})
        S = 0
T = shortest path tree of Q
```

Figure 4: Computing a bounded-radius spanning tree T for $G = (V, E)$, with source $s \in V$, using parameter ϵ. T will have cost at most $(1 + \frac{1}{\epsilon})$ times the cost of MST_G, and radius at most $(1 + \epsilon)$ times R.

We now prove that for any fixed ϵ this algorithm produces a routing tree with radius and total cost simultaneously bounded by small constants times optimum:

Theorem 1: For any weighted graph G and parameter ϵ, the routing tree T constructed by our algorithm has $cost(T) \leq (1 + \frac{1}{\epsilon}) \cdot cost(MST_G)$.

Proof: Let $v_1, v_2, \ldots, v_k$ be the set of nodes to which the algorithm added shortest paths from the source node, and let $v_0 = s$. By the algorithm construction we have

$$cost(T) \leq cost(MST_G) + \sum_{i=1}^{k} dist_G(s, v_i)$$

$$\leq cost(MST_G) + \sum_{i=1}^{k} \frac{1}{2\epsilon} \cdot dist_L(v_{i-1}, v_i)$$

$$\leq cost(MST_G) + \frac{1}{2\epsilon} \cdot cost(L)$$

Since $cost(L) \leq 2 \cdot cost(MST_G)$, we have

$$cost(T) \leq cost(MST_G) + \frac{1}{\epsilon} \cdot cost(MST_G)$$

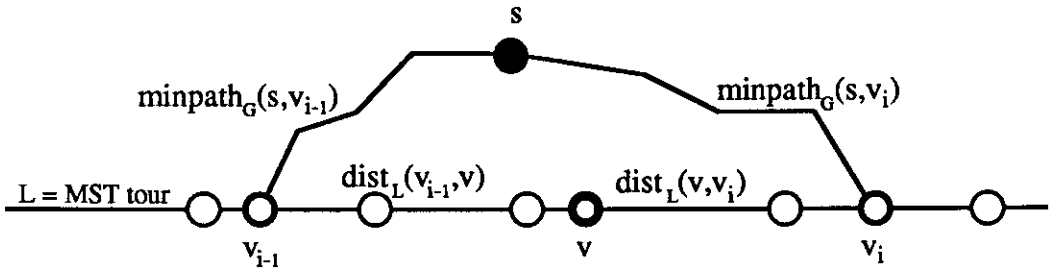
$$= (1 + \frac{1}{\epsilon}) \cdot cost(MST_G)$$

□

Theorem 1 suggests that for $\epsilon = 0$, the ratio $\frac{cost(T)}{cost(MST_G)}$ is not bounded by any constant, and indeed this is true for the example of Figure 1 above, where $\frac{cost(T)}{cost(MST_G)}$ is $\Omega(|V|)$.

Theorem 2: For any weighted graph G and parameter ϵ, the routing tree T constructed by our algorithm has radius $r(T) \leq (1 + \epsilon) \cdot R$.

Proof: For any $v \in V$, let v_{i-1} be the last node before v on the MST tour L for which we added $minpath_G(s, v_{i-1})$ to E' in the algorithm, and similarly let v_i be the node following v on L for which we added $minpath_G(s, v_i)$ to E'.



By the construction of the algorithm, we know that

$$dist_L(v_{i-1}, v) + dist_L(v, v_i) \leq 2\epsilon \cdot dist_G(s, v_i) \leq 2\epsilon \cdot R$$

and without loss of generality we may assume that $dist_L(v_{i-1}, v) \leq \frac{2\epsilon \cdot R}{2} = \epsilon \cdot R$. We then have

$$dist_T(s, v) \leq dist_T(s, v_{i-1}) + dist_L(v_{i-1}, v)$$

$$\leq dist_G(s, v_{i-1}) + \epsilon \cdot R$$

$$\leq R + \epsilon \cdot R = (1 + \epsilon) \cdot R$$

□

A similar idea was recently used in the distributed computation literature by Awerbuch and coworkers [1] for constructing spanning trees with small diameter and small weight. However, our algorithm treats the bounded-radius minimum spanning tree problem, while they treat the tree diameter instead. Moreover, our method involves a simpler construction with tighter performance bounds.

4 Bounded Radius Steiner Tree Global Routing

In the previous section, we treated the bounded radius minimum spanning tree problem, where each net N is routed in an underlying routing graph $G = (V, E)$ with $V = N$. As noted above, the spanning tree routing has proved useful in cell-based design. However, for building-block design the underlying routing graph is based on the channel intersection graph [4], and a net N is a subset of the vertices of G.

In this case, the BRMRT problem is actually the **Bounded Radius Optimal Steiner Tree** (BROST) problem, and the channel intersection points (i.e., the nodes in $V - N$) are potential Steiner points. The following is immediate:

Lemma: The BROST problem is NP-complete.

Proof: Setting $\epsilon = \infty$ yields the graph Steiner problem, which is known to be NP-complete [10]. □

Our approximation algorithm for the bounded radius optimal Steiner tree problem is similar to the algorithm presented in Section 3. Note that given any approximate Steiner tree $\hat{T}$, we can use the approach of Section 3 to construct a routing tree with cost within $(1 + \frac{1}{\epsilon}) \cdot cost(\hat{T})$, and with radius within $(1 + \epsilon) \cdot r(\hat{T})$. Our algorithm uses a heuristic of Kou, Markovsky and Berman (KMB) [16] to build a Steiner tree in the underlying routing graph having cost within a factor 2 of optimal.[2]

[2]Given a graph $G = (V, E)$ and a net of terminals $N \subseteq V$, the method of Kou, Markovsky and Berman is as follows.

We construct the depth-first tour of the heuristic Steiner tree. Next, we traverse the tour, adding to MST_G the shortest paths from the source to the appropriate vertices of the tour, as in Section 3. Finally, we compute the shortest path tree in the resulting graph and output the union of the shortest paths from the source to all terminals in N (which includes intermediate non-terminal nodes on the shortest paths as Steiner points). Note that the cost of the tour will be at most 4 times the *optimal* Steiner tree (T_{opt}) cost. Thus, the resulting routing tree cost is at most $2 \cdot (1 + \frac{1}{\epsilon})$ times optimal.

Theorem 3: For any weighted graph $G = (V, E)$, node subset $N \subseteq V$ and parameter ϵ, the routing tree T constructed by our algorithm has $cost(T) \leq 2 \cdot (1 + \frac{1}{\epsilon}) \cdot cost(T_{opt})$, and radius $r(T) \leq (1+\epsilon) \cdot R$.

Proof: By our previous arguments, $r(T) \leq (1 + \epsilon) \cdot R$. In addition, $cost(T) \leq (1 + \frac{1}{\epsilon}) \cdot cost(\hat{T})$, where $\hat{T}$ is the approximate Steiner tree produced by the method of [16]. Since $cost(\hat{T}) \leq 2 \cdot cost(T_{opt})$, we have $cost(T) \leq 2 \cdot (1 + \frac{1}{\epsilon}) \cdot cost(T_{opt})$. □

Well-known results which bound the $\frac{MST}{Steiner}$ ratio in various metrics can be combined with Theorem 3 to yield even better bounds whenever the edge weights correspond to a metric (e.g., Manhattan or Euclidean). To illustrate this, we give two immediate examples:

Corollary 1: Given a set of terminals N in the Manhattan plane and a real parameter ϵ, our algorithm will produce a routing tree T with cost bounded by $\frac{3}{2} \cdot (1 + \frac{1}{\epsilon})$ times optimal, and $r(T)$ bounded by $(1 + \epsilon)$ times optimal.

Proof: By a result of Hwang [9], the rectilinear minimum spanning tree gives a $\frac{3}{2}$ approximation to the optimal Steiner tree. We then apply an argument similar to that of Theorem 3. □

Corollary 2: Given a set of terminals N in the Euclidean plane and a real parameter ϵ, our algorithm will produce a routing tree T with cost bounded by $\frac{2}{\sqrt{3}} \cdot (1 + \frac{1}{\epsilon})$ times optimal, and $r(T)$ bounded by $(1 + \epsilon)$ times optimal.

Proof: By a recent result of Du and Hwang [7], the Euclidean minimum spanning tree gives a $\frac{2}{\sqrt{3}}$ approximation to the optimal Steiner tree. We again apply the basic argument of Theorem 3. □

First, construct the complete graph over N with each edge weight equal to the cost of the corresponding shortest path in G. Compute T, the minimum spanning tree of this complete graph, and expand each edge of T into the corresponding shortest path, yielding a subgraph G' that spans N. Finally, compute the minimum spanning tree T' of G', and delete edges from T' until all leaves are nodes of N. Output the resulting tree.

5 Experimental Results

Both approximation algorithms, for bounded radius minimum spanning tree routing and for bounded radius optimal Steiner tree routing, were implemented in ANSI C for the Sun-4, Macintosh and IBM environments; code is available from the authors upon request. The algorithms were respectively tested on a large number of random pointsets, and on channel intersection graphs of random block placements. These are standard testbeds which capture the statistical properties of signal nets in actual layouts.

The bounded radius minimum spanning tree algorithm was tested on random pointsets generated from a uniform distribution in the grid. Results are summarized in Figure 6, which clearly shows the tradeoff between routing cost and maximum delay. As ϵ decreases, both the cost and radius curves shift monotonically from that of the minimum spanning tree to that of the shortest path tree. More detailed data is given by Table 1 in Appendix I.

The approximation algorithm for the bounded radius optimal Steiner tree problem was tested on random block layouts in the grid; these were generated by adding a fixed number of non-overlapping blocks, with length, width and lower-left coordinates all uniformly distributed. Given a block design, nets with terminals on the block peripheries were routed within the corresponding channel intersection graph. An example of the output from our algorithm is shown in Figure 5.

A detailed summary of experimental results for Steiner routing in block designs is contained in Table 2 of Appendix I. Once again, the simulations confirm the tradeoffs inherent in the bounded-radius routing approach. Note that although our construction starts with the heuristic Steiner tree of Kou, Markovsky and Berman, our routing solution may in some cases have smaller cost than the KMB tree. In all cases, the radius of our routing tree is no larger than that of the KMB tree. This too is reflected in the experimental data.

6 Conclusions

We propose a new, provably good general algorithm for spanning tree global routing in arbitrary design methodologies ranging from cell-based to building-block design. Our global router is based on

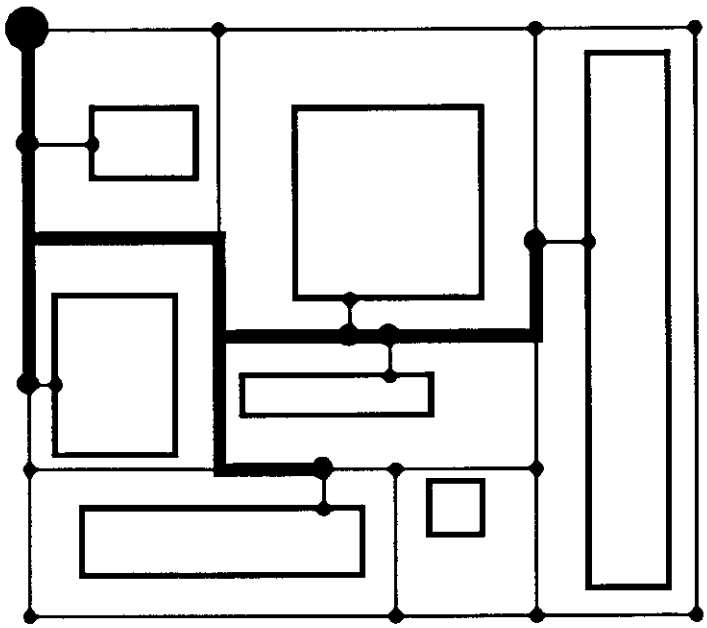


Figure 5: A set of placed modules and their channel intersection graph. The highlighted tree is the routing produced by our algorithm.

a routing tree construction where *both* the total wirelength and the maximum delay of the routing are bounded by *constant* factors away from optimal, settling an open problem proposed in [2]. Our approach readily extends to Steiner tree routing in arbitrary weighted graphs, where again the routing tree is only a small constant factor away from optimal in terms of both cost and radius. Extensive simulations over geometric routing graphs as well as channel intersection graphs derived from random block designs confirm that our approach gives very good performance. The results of Section 5 indeed exhibit a smooth tradeoff between the competing requirements of minimum delay and minimum total wirelength.

Based on our methods for constructing bounded radius routing trees, the global routing procedure will work as follows. We route all nets, one by one, according to their priorities. For each net, we construct a bounded radius minimum spanning tree or bounded radius minimum Steiner tree using the algorithms presented in Sections 3 and 4. The parameter ϵ is either given by the user or computed based on an estimation of the timing constraints for the net. The cost of each edge in the routing graph is a function of wirelengths, channel capacities, and the distribution of current channel densities. After routing each net, we update the edge costs in the routing graph. After all nets are routed, we

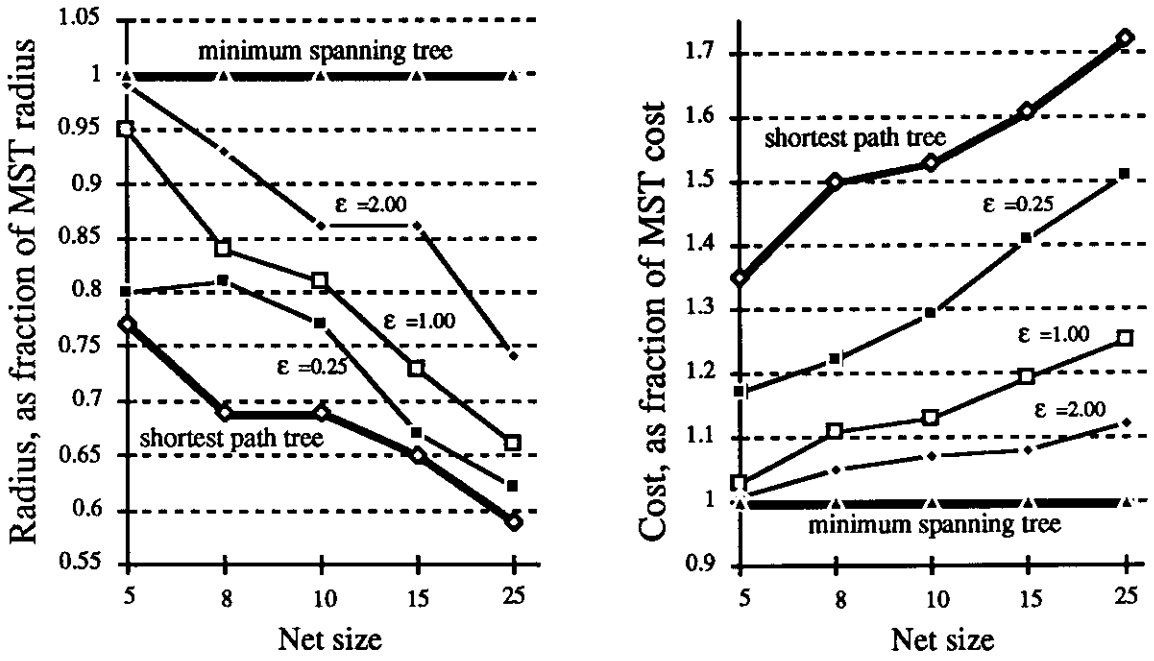


Figure 6: These charts illustrate the smooth tradeoff induced by our algorithm between total routing cost and maximum signal delay. In both cases the envelope of performance lies between the shortest path tree and the minimum spanning tree, and the parameter ϵ determines the exact tradeoff.

may compute the timing-critical paths and, if necessary, further reduce the interconnection delay by re-routing some critical nets based on more accurate distributed RC delay models.

Our algorithms readily extend to other norms and to alternate geometries (e.g., 45- or 30-60-90-degree routing regimes). There are several remaining open problems, such as the complexity of the bounded radius minimum spanning tree problem for points in the plane, or how to choose an MST with minimum radius when the MST is not unique.

References

[1] B. Awerbuch, A. Baratz and D. Peleg, "Cost-Sensitive Analysis of Communication Protocols", *Proc. ACM Symp. on Principles of Distributed Computing*, 1990, pp. 177-187.

[2] J. Cong, A. Kahng, G. Robins, M. Sarrafzadeh,and C. K. Wong, "Performance-Driven Global Routing for Cell Based IC's", to appear in *Proc. IEEE Intl. Conf. on Computer Design*, 1991.

[3] J. Cong and B. Preas, "A New Algorithm for Standard Cell Global Routing", *Proc. IEEE Intl. Conf. on Computer Aided Design*, November 1988, pp. 176-179.

[4] W. M. Dai, T. Asano and E. S. Kuh, "Routing Region Definition and Ordering Scheme for Building-Block Layout", *IEEE Trans. on CAD*, Vol. CAD-4, July 1985, pp. 189-197.

[5] W. E. Donath, R. J. Norman, B. K. Agrawal and S. E. Bello, "Timing Driven Placement Using Complete Path Delays", *Proc. IEEE Design Automation Conf.*, 1990, pp. 84-89.

[6] A. E. Dunlop, V. D. Agrawal, D.N. Deutsch, M. F. Jukl, P. Kozak and M. Wiesel, "Chip Layout Optimization Using Critical Path Weighting", *Proc. IEEE Design Automation Conf.*, 1984, pp. 133-136.

[7] D. Z. Du and F. K. Hwang, "A Proof of Gilbert-Pollak's Conjecture on the Steiner Ratio", *Proc. IEEE Symp. on Foundations of Computer Science*, 1990.

[8] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers", *J. Appl. Phys.* 19(1) (1948), pp. 55-63.

[9] F. K. Hwang, "On Steiner Minimal Trees with Rectilinear Distance", *SIAM J. of Applied Math.* 30(1) (1976), pp. 104-114.

[10] M. R. Garey and D. S. Johnson, "Computers and Intractability: a Guide to the Theory of NP Completeness", W. H. Freeman, San Francisco, 1979.

[11] P. S. Hauge, R. Nair and E. J. Yoffa, "Circuit Placement for Predictable Performance", *Proc. IEEE Intl. Conf. on Computer Aided Design*, 1987, pp. 88-91.

[12] J. Ho, D. T. Lee, C. H. Chang and C. K. Wong, "Bounded-Diameter Spanning Trees and Related Problems", *Proc. ACM Symp. on Computational Geometry*, 1989, pp. 276-282.

[13] M. A. B. Jackson and E. S. Kuh, "Performance-Driven Placement of Cell-Based IC's", *Proc. IEEE Design Automation Conf.*, 1989, pp. 370-375.

[14] M. A. B. Jackson, A. Srinivasan and E. S. Kuh, "Clock Routing for High-Performance IC's", *Proc. IEEE Design Automation Conf.*, 1990, pp. 573-579.

[15] M. A. B. Jackson, E. S. Kuh and M. Marek-Sadowska, "Timing-Driven Routing for Building Block Layout", *Proc. IEEE Intl. Symp. on Circuits and Systems*, 1987, pp. 518-519.

[16] L. Kou, G. Markowsky, and L. Berman "A Fast Algorithm for Steiner Trees", *Acta Informatica* 15, 1981, pp. 141-145.

[17] I. Lin and D. H. C. Du, "Performance-Driven Constructive Placement", *Proc. IEEE Design Automation Conference*, 1990, pp. 103-106.

[18] M. Marek-Sadowska and S. P. Lin, "Timing Driven Placement", *Proc. IEEE Intl. Conf. on Computer Aided Design*, 1989, pp. 94-97.

[19] Y. Ogawa, M. Pedram and E. S. Kuh, "Timing-Driven Placement for General Cell Layout", *Proc. International Symp. on Circuits and Systems*, 1990, pp. 872-876.

[20] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Prentice-Hall, 1982.

[21] P. Ramanathan and K. G. Shin, "A Clock Distribution Scheme for Non-Symmetric VLSI Circuits", *Proc. IEEE Intl. Conf. on Computer Aided Design*, 1989, pp. 398-401.

[22] J. Rubinstein, P. Penfield and M. A. Horowitz, "Signal Delay in RC Tree Networks", *IEEE Trans. on CAD* 2(3) (1983), pp. 202-211.

[23] S. Sutanthavibul and E. Shragowitz, "An Adaptive Timing-Driven Layout for High Speed VLSI", *Proc. IEEE Design Automation Conf.*, 1990, pp. 90-95.

7 Appendix I: Experimental Results

7.1 Data for Random Nets

The following table gives minimum, maximum, and average values of the routing tree radius and the SPT radius, as well as the routing tree cost and the shortest path tree cost, all represented as fractions of the corresponding values for the minimum spanning tree. The data shown represents averages over 50 random pointsets for each cardinality. The points were independently and uniformly distributed in the unit square, and the source was selected to be one of the terminals at random.

ϵ	net size	tree radius of our algorithm			Shortest Path tree radius			tree cost of our algorithm			Shortest Path tree cost		
		min	ave	max	min	ave	max	min	ave	max	min	ave	max
0.01	5	0.55	0.85	1.00	0.55	0.85	1.00	1.00	1.29	1.86	1.00	1.29	1.86
0.01	8	0.40	0.77	1.00	0.40	0.77	1.00	1.02	1.38	2.33	1.02	1.38	2.33
0.01	10	0.33	0.74	1.00	0.33	0.74	1.00	1.04	1.48	2.28	1.04	1.48	2.28
0.01	15	0.37	0.69	0.95	0.37	0.69	0.95	1.15	1.50	2.02	1.15	1.50	2.02
0.01	25	0.25	0.60	0.97	0.25	0.60	0.97	1.34	1.72	2.66	1.34	1.72	2.66
0.10	5	0.44	0.82	1.00	0.44	0.81	1.00	1.00	1.25	1.84	1.00	1.30	1.96
0.10	8	0.43	0.74	1.00	0.43	0.74	1.00	1.03	1.35	1.99	1.03	1.41	1.99
0.10	10	0.38	0.71	1.00	0.38	0.70	1.00	1.00	1.39	1.96	1.05	1.45	2.25
0.10	15	0.27	0.65	1.00	0.27	0.65	1.00	1.20	1.53	2.66	1.20	1.60	2.71
0.10	25	0.34	0.64	0.94	0.34	0.63	0.93	1.25	1.57	2.03	1.25	1.66	2.16
0.25	5	0.47	0.80	1.00	0.47	0.77	1.00	1.00	1.17	1.72	1.00	1.30	2.25
0.25	8	0.40	0.81	1.00	0.39	0.78	1.00	1.00	1.22	1.82	1.00	1.38	1.91
0.25	10	0.38	0.77	1.00	0.38	0.74	1.00	1.01	1.29	1.72	1.04	1.47	2.08
0.25	15	0.39	0.67	0.95	0.39	0.65	0.95	1.13	1.41	2.05	1.19	1.58	3.18
0.25	25	0.37	0.62	0.88	0.37	0.60	0.85	1.21	1.51	1.86	1.30	1.72	2.28
0.50	5	0.57	0.90	1.00	0.47	0.85	1.00	1.00	1.15	1.60	1.00	1.25	2.04
0.50	8	0.48	0.74	0.99	0.46	0.69	0.99	1.00	1.22	1.66	1.02	1.37	1.94
0.50	10	0.42	0.81	1.00	0.37	0.75	1.00	1.00	1.23	1.57	1.02	1.45	2.05
0.50	15	0.44	0.72	0.99	0.42	0.69	0.99	1.11	1.29	1.53	1.13	1.54	1.94
0.50	25	0.33	0.66	0.97	0.31	0.63	0.97	1.17	1.34	1.73	1.32	1.60	2.14
0.75	5	0.58	0.87	1.00	0.40	0.78	1.00	1.00	1.10	1.44	1.00	1.26	1.90
0.75	8	0.52	0.77	1.00	0.45	0.70	1.00	1.00	1.17	1.40	1.11	1.40	2.15
0.75	10	0.51	0.78	1.00	0.44	0.71	1.00	1.00	1.20	1.54	1.09	1.53	2.34
0.75	15	0.45	0.72	1.00	0.39	0.67	0.94	1.03	1.23	1.66	1.11	1.54	2.40
0.75	25	0.36	0.64	0.98	0.34	0.59	0.92	1.08	1.29	1.58	1.23	1.68	2.40
1.00	5	0.66	0.95	1.00	0.56	0.83	1.00	1.00	1.03	1.30	1.00	1.22	1.90
1.00	8	0.56	0.84	1.00	0.44	0.74	0.95	1.00	1.11	1.31	1.12	1.37	1.96
1.00	10	0.51	0.81	1.00	0.50	0.73	1.00	1.00	1.13	1.47	1.03	1.45	1.96
1.00	15	0.44	0.73	1.00	0.30	0.66	0.97	1.00	1.19	1.41	1.14	1.61	2.28
1.00	25	0.32	0.66	0.99	0.31	0.61	0.93	1.11	1.25	1.43	1.37	1.71	2.38
2.00	5	0.84	0.99	1.00	0.50	0.78	1.00	1.00	1.01	1.15	1.00	1.30	2.03
2.00	8	0.47	0.93	1.00	0.40	0.72	0.94	1.00	1.05	1.22	1.03	1.48	2.06
2.00	10	0.51	0.86	1.00	0.38	0.69	1.00	1.00	1.07	1.23	1.10	1.50	2.28
2.00	15	0.48	0.86	1.00	0.35	0.69	1.00	1.00	1.08	1.19	1.18	1.49	1.95
2.00	25	0.36	0.74	1.00	0.23	0.60	1.00	1.01	1.12	1.27	1.25	1.68	2.31
3.00	5	0.80	0.99	1.00	0.45	0.77	1.00	1.00	1.00	1.10	1.00	1.35	2.84
3.00	8	0.57	0.96	1.00	0.44	0.71	1.00	1.00	1.01	1.12	1.04	1.50	2.46
3.00	10	0.69	0.96	1.00	0.50	0.73	1.00	1.00	1.03	1.17	1.12	1.48	2.51
3.00	15	0.44	0.91	1.00	0.36	0.70	0.98	1.00	1.05	1.13	1.07	1.56	2.12
3.00	25	0.40	0.81	1.00	0.30	0.61	0.89	1.00	1.08	1.22	1.34	1.67	2.40

Table 1: Heuristic tree and shortest path tree radius and cost statistics for random pointsets, expressed as a fraction of the corresponding minimum spanning tree values.

7.2 Channel Intersection Graphs for Random Block Designs

The following table gives minimum, maximum, and average values of the routing tree radius and the SPT radius, as well as the routing tree cost and the shortest path tree cost, all represented as fractions of the corresponding values for the heuristic tree of Kou, Markovsky and Berman. The data shown represents averages over 50 sets of 15 randomly placed modules, with random sets of terminals being routed within the channel intersection graph. Again, the source node was selected to be one of the terminals at random.

ϵ	net size	tree radius of our algorithm			Shortest Path tree radius			tree cost of our algorithm			Shortest Path tree cost		
		min	ave	max	min	ave	max	min	ave	max	min	ave	max
0.10	3	0.63	0.93	1.00	0.63	0.93	1.00	0.91	1.12	1.42	0.91	1.12	1.42
0.10	4	0.50	0.90	1.00	0.50	0.90	1.00	0.96	1.14	1.69	0.96	1.14	1.69
0.10	5	0.43	0.84	1.00	0.43	0.84	1.00	0.99	1.17	1.51	0.99	1.18	1.57
0.10	7	0.42	0.82	1.00	0.42	0.82	1.00	0.99	1.14	1.45	0.99	1.15	1.47
0.10	10	0.51	0.82	1.00	0.51	0.82	1.00	1.00	1.22	1.58	1.00	1.22	1.58
0.10	15	0.31	0.81	1.00	0.31	0.80	1.00	1.02	1.21	1.53	1.02	1.22	1.53
0.25	3	0.55	0.89	1.00	0.55	0.89	1.00	0.95	1.14	1.49	0.95	1.17	1.73
0.25	4	0.59	0.94	1.00	0.59	0.93	1.00	1.00	1.10	1.52	1.00	1.14	1.52
0.25	5	0.51	0.90	1.00	0.51	0.88	1.00	1.00	1.11	1.56	1.00	1.15	1.59
0.25	7	0.53	0.86	1.00	0.53	0.85	1.00	0.98	1.18	1.72	0.98	1.23	1.83
0.25	10	0.58	0.83	1.00	0.58	0.83	1.00	1.00	1.15	1.45	1.01	1.18	1.56
0.25	15	0.46	0.77	1.00	0.46	0.75	1.00	1.00	1.16	1.34	1.03	1.19	1.48
0.50	3	0.60	0.94	1.00	0.60	0.94	1.00	0.89	1.09	1.57	1.00	1.14	1.61
0.50	4	0.55	0.88	1.00	0.52	0.86	1.00	0.98	1.11	1.43	1.00	1.16	1.51
0.50	5	0.43	0.89	1.00	0.43	0.87	1.00	0.97	1.15	1.68	0.97	1.23	1.61
0.50	7	0.48	0.86	1.00	0.45	0.82	1.00	0.96	1.11	1.41	0.96	1.20	1.61
0.50	10	0.42	0.80	1.00	0.42	0.77	1.00	0.98	1.17	1.50	1.00	1.27	1.58
0.50	15	0.40	0.78	1.00	0.40	0.75	1.00	0.96	1.15	1.41	0.96	1.19	1.51
0.75	3	0.57	0.96	1.00	0.54	0.92	1.00	1.00	1.06	1.40	0.96	1.17	1.57
0.75	4	0.66	0.95	1.00	0.51	0.90	1.00	1.00	1.05	1.29	1.00	1.16	1.75
0.75	5	0.52	0.93	1.00	0.51	0.90	1.00	0.98	1.06	1.33	1.00	1.19	1.57
0.75	7	0.29	0.86	1.00	0.29	0.81	1.00	0.99	1.10	1.35	1.00	1.17	1.51
0.75	10	0.54	0.84	1.00	0.49	0.79	1.00	0.98	1.13	1.37	1.02	1.27	1.69
0.75	15	0.42	0.82	1.00	0.42	0.78	1.00	1.01	1.14	1.41	1.00	1.21	1.76
1.00	3	0.65	0.99	1.00	0.57	0.93	1.00	0.89	1.02	1.27	0.89	1.14	1.72
1.00	4	0.64	0.99	1.00	0.54	0.91	1.00	0.97	1.02	1.19	1.00	1.15	1.71
1.00	5	0.67	0.95	1.00	0.48	0.86	1.00	1.00	1.09	1.38	1.00	1.23	1.87
1.00	7	0.55	0.92	1.00	0.55	0.84	1.00	1.00	1.09	1.37	1.00	1.21	1.58
1.00	10	0.53	0.90	1.00	0.47	0.81	1.00	0.96	1.10	1.32	1.01	1.22	1.69
1.00	15	0.47	0.85	1.00	0.47	0.78	1.00	0.98	1.11	1.30	0.97	1.21	1.71
2.00	3	1.00	1.00	1.00	0.62	0.92	1.00	1.00	1.00	1.00	1.00	1.13	1.51
2.00	4	0.71	0.99	1.00	0.55	0.89	1.00	0.92	1.01	1.26	0.92	1.15	1.59
2.00	5	0.61	0.99	1.00	0.59	0.85	1.00	1.00	1.02	1.23	1.00	1.19	1.64
2.00	7	0.49	0.97	1.00	0.43	0.80	1.00	0.95	1.03	1.22	0.97	1.22	1.59
2.00	10	0.49	0.93	1.00	0.45	0.81	1.00	1.00	1.05	1.25	1.02	1.26	1.75
2.00	15	0.46	0.88	1.00	0.45	0.76	1.00	0.99	1.06	1.21	1.06	1.25	1.49
3.00	3	1.00	1.00	1.00	0.58	0.92	1.00	1.00	1.00	1.00	1.00	1.14	1.79
3.00	4	0.76	1.00	1.00	0.59	0.88	1.00	1.00	1.00	1.13	0.97	1.14	1.68
3.00	5	1.00	1.00	1.00	0.50	0.85	1.00	1.00	1.01	1.12	0.91	1.15	1.50
3.00	7	0.76	0.98	1.00	0.49	0.85	1.00	1.00	1.01	1.11	1.00	1.23	1.83
3.00	10	0.73	0.98	1.00	0.50	0.78	1.00	0.97	1.01	1.11	1.00	1.23	1.60
3.00	15	0.48	0.96	1.00	0.44	0.77	1.00	0.99	1.03	1.17	0.98	1.20	1.51

Table 2: Heuristic tree and shortest path tree radius and cost statistics for random block designs, expressed as fractions of the corresponding KMB tree values.