Fast Optimal Algorithms for the Minimum Rectilinear Steiner Arborescence Problem

Kwok-Shing Leung
Advanced Technology Group
Cadence Design Systems, San Jose, CA 95134

Jason Cong
Computer Science Department
University of California, Los Angeles, CA 90095

Abstract— **In this paper, we present two optimal algorithms for solving the Minimum Rectilinear Steiner Arborescence (MRSA) Problem. The first algorithm is a recursive branch-and-bound variant of the RSA heuristic by Rao *et al.* [14]. The second algorithm uses dynamic programming to avoid solving the same subproblem more than once. Furthermore, both algorithms can be generalized to solve the All-Quadrant MRSA Problem. Extensive experimental results show that our algorithms significantly outperform existing exact methods for solving the MRSA problem.**

I. Introduction

Given a set of terminals $S = \{S_1, S_1, S_2, \cdots, S_n\}$ such that $S_1 = (0, 0)$ is the source, and $S - \{S_1\}$ is the set of sinks all located in the first quadrant ($x_i \geq 0, y_i \geq 0, \forall 1 \leq i \leq n$). The Minimal Rectilinear Steiner Arborescence (MRSA) Problem is to construct a rectilinear Steiner tree T with the minimum total edge length subject to the constraint that for all $1 \leq i \leq n$, the path from S_1 to S_i in T is a rectilinear shortest path. The MRSA problem was first studied by Ladeira de Matos [10] and Nastansky *et al.* [12]. Ho *et al.* [7] presented two exponential algorithms for the MRSA problem with runtime $O(n^2 3^n)$ and $O(n^{3k})$, respectively (where k is the number of dominating layers, which is $\Omega(n)$ in the worst case). Recently Rao *et al.* [14] presented the RSA heuristic which constructs an RSA in $O(n \log n)$ time, whose length has an upper bound of twice that of the MRSA. Cong *et al.* [3] presented the heuristic A-tree algorithm based on making optimal safe moves which runs in $O(n^3)$ time. An optimal branch-and-bound variant of the A-tree algorithm was also described in a subsequent work [4]. Alexander and Robins [1] presented the IDOM heuristic adapted from the well-known Iterated 1-Steiner heuristic for the rectilinear Steiner tree problem [8]. Córdova and Lee [6] proposed a straight-forward generalization of the RSA heuristic to the all-quadrant MRSA problem (the location of the sinks are not restricted to be in the first quadrant), with the same runtime complexity and performance bound as the RSA heuristic. Finally, Téllez and Sarrafzadeh [15] proposed another heuristic which runs slightly better than the RSA heuristic on average.

With the advent of the VLSI design and process technology, the MRSA problem has received much attention in the field of VLSI computer-aided design. Cong *et al.* [3] showed that, with the current process technology, rectilinear Steiner arborescence has significant advantage over the tradition Steiner tree approach in delay optimization. Cong and Madden [5] proposed a multi-source routing algorithm based on the construction of minimum-cost minimum-diameter arborescences. Okamoto and Cong [13] extended the RSA heuristic to perform simultaneous (arborescence-based) topology optimization and buffer insertion. Alexander and Robins [1] used Steiner arborescence in the general graph to route critical nets in FPGA. The MRSA problem also has applications in the message routing in multicomputer networks [2], [7].

In this paper, we present two optimal algorithms for solving the MRSA problem. The first algorithm is a recursive branch-and-bound variant of the RSA heuristic. The second algorithm uses dynamic programming to avoid solving the same subproblem more than once. In particular, we show that each subproblem in the recursion can be completely characterized by a triple (P, K, C), which can be defined and solved recursively. We use hashing-based dynamic programming to limit the total number of triples (subproblems) solved, and such technique was proven to be highly effective. Furthermore, both algorithms can be generalized to solve the All-Quadrant MRSA Problem. Extensive experimental results show that our dynamic-programming-based algorithm can solve the single-quadrant MRSA problem with 100 terminals optimally in one minute, and the all-quadrant MRSA problem with 240 terminals optimally in one hour. Our algorithms significantly outperform existing exact methods for solving the MRSA problem, which can only handle 20 to 35 nodes in an hour. Due to the length limitation, some theorems and details of the algorithms are left out in this paper. The reader may refer to [11] for details.

The following notations are used in this paper. Let p, q be two generic points in the first quadrant of the Manhattan plane. We use (x_p, y_p) and (x_q, y_q) to denote their coordinates, respectively. We define $\langle p, q \rangle$ as the point with coordinates $(\min\{x_p, x_q\}, \min\{y_p, y_q\})$. We use $|p|$ as a shorthand for $|x_p| + |y_p|$. We say q is *dominated* by p, or $q \prec p$, if and only if $x_q \leq x_p$, $y_q \leq y_p$, and $p \neq q$. If $q \prec p$, $q \rightsquigarrow p$ denotes a Manhattan shortest path connection from q to p (pointing outward from q to p), and $|q \rightsquigarrow p| = |p| - |q|$ denotes its length. Given a set P and a node m, $P_{\succ}(m)$ denotes the set $\{p \mid p \in P \text{ and } m \prec p\}$, which is the subset of P dominating m. Note that any arborescence can be partitioned into a set of paths of the form $q \rightsquigarrow p$, where q is called the (unique) *parent* of p, and p is called a *child* of q. Given an integer $K \geq 0$, the equation $x + y = K$ is called the *K-scan*, and K is called the *scan level*.

II. Review of The RSA Heuristic

The RSA heuristic [14] constructs an arborescence in a bottom-up fashion, starting with n subtrees each consisting

This work was partially supported by the NSF Young Investigator Award MIP-9357582, and a grant from Intel Corporation. Kwok-Shing Leung was with Intel Corporation, Hillsboro, OR 97124, when this work was performed.

of a terminal in S. It iteratively *merges* pairs of roots of subtrees p and q such that $\langle p, q\rangle$ is as far from the source as possible, and terminates when there is only one subtree left. In our implementation, the heuristic maintains the *peer set* P consisting of all the roots of current subtrees above a particular scan level K, Starting with $K = \infty$ and $P = \phi$, the RSA heuristic iteratively finds the next highest scan level K' such that one of the following two cases happen:

- CASE I (TERMINAL MERGER) – $\exists\ m \in S - P$ such that $|m| = K'$.
- CASE II (STEINER MERGER) – $\exists\ m_N, m_E \in P\ (m_N \neq m_E)$ such that $m = \langle m_N, m_E\rangle \notin S$ and $|m| = K'$.

In CASE I (TERMINAL MERGER), m is called the *terminal merging point*, and the RSA heuristic adds an edge from m to each of the nodes in $P_{\succ}(m)$. In CASE II (STEINER MERGER), m is called the *Steiner merging point*, and two edges $m \rightsquigarrow m_N$ and $m \rightsquigarrow m_E$ will be added to the partial arborescence A ($A = \phi$ initially):

- ACTION I –
  - $A \leftarrow A + \{m \rightsquigarrow p \mid p \in P_{\succ}(m)\}$
  - $P \leftarrow P - P_{\succ}(m) + \{m\}$
- ACTION II –
  - $A \leftarrow A + \{m \rightsquigarrow m_N, m \rightsquigarrow m_E\}$
  - $P \leftarrow P - \{m_N, m_E\} + \{m\}$

The algorithm terminates when $K = 0$, and returns the the arborescence A.

III. Branch-and-Bound Algorithm

The RSA heuristic works by iteratively merging at the furthest possible merging point. We can show that if the current partial arborescence A is *optimal* (i.e. there exists an optimal arborescence A_{opt} such that $A \subseteq A_{opt}$), then performing a terminal merger guarantees that the resulting partial arborescence is also optimal (see [11] for details). Unfortunately, there is no guarantee that Steiner mergers preserves optimality of the partial arborescence. In fact, Kahng and Robins gave an example for which the length of the arborescence computed by the RSA heuristic is arbitrarily close to twice the optimal value [9].

In general, we can construct better arborescences by selectively *forbidding* Steiner mergers at specific locations. With this observation, we developed the RSA/BnB algorithm, which is an optimal branch-and-bound variant of the RSA heuristic. As in the RSA heuristic, RSA/BnB starts with $P = \phi$ and $K = \infty$, and iteratively finds the next highest scan level K' such that one of the two cases happen. However, instead of always choosing to merge in CASE II, RSA/BnB tries out *both* merging and skipping:

- ACTION I – same action as before:
  - $A \leftarrow A + \{m \rightsquigarrow p \mid p \in P_{\succ}(m)\}$
  - $P \leftarrow P - P_{\succ}(m) + \{m\}$
- ACTION II – two branches will be created:
  - ⋆ ACTION II.MERGE – in the first branch, m_N and m_E are merged into $\langle m_N, m_E\rangle$:
    - $A \leftarrow A + \{m \rightsquigarrow m_N, m \rightsquigarrow m_E\}$
    - $P \leftarrow P - \{m_N, m_E\} + \{m\}$
  - ⋆ ACTION II.SKIP – in the second branch, A and P remain unchanged.

We can show that at least one leaf node in the branch-and-bound diagram corresponds to an optimal arborescence, and therefore RSA/BnB is optimal. Again, the reader may refer to [11] for more details regarding the algorithm, the optimality proof, as well as other pruning techniques.

IV. Dynamic Programming

While RSA/BnB is more efficient than other previous approaches (as we will see in the following sections), the exponential behavior of the branch-and-bound approach quickly becomes a limiting factor as the size of the input increases. The main reason is that the same subproblem may be recomputed over and over again in many different branches.

However, we observe that each subproblem (node in the branch-and-bound diagram) can be completely characterized by the triple (P, K, C), where P is the peer set, K is the scan level, and C is the cost of the partial arborescence constructed so far. With such characterization in mind, we can avoid solving the same subproblem more than once using dynamic programming, and our new optimal algorithm is called RSA/DP. In particular, instead of solving the triple (P, K, C) recursively, we expand the triple into zero, one, or two child triples (corresponding to the node being pruned, ACTION I, and ACTION II, respectively) to be solved later:

- ACTION I – create $(P - P_{\succ}(m) + \{m\}, |m|, C + \sum_{p \in P_{\succ}(m)} |m \rightsquigarrow p|)$.
- ACTION II – create $(P - \{m_N, m_E\} + \{m\}, |m|, C + |m \rightsquigarrow m_N| + |m \rightsquigarrow m_E|)$ corresponding to merging, and $(P, |m|, C)$ corresponding to skipping.

Since a triple only generates triples with a smaller scan level K', we should expand all the triples with the highest scan level before proceeding to the next scan level. By doing so, it is guaranteed that once we have expanded a particular triple (P, K, C), the same triple would not be regenerated during a later expansion of some other triples and so the triple can be deleted immediately (provided that the problem is "z-distinct", see [11] for details). Moreover, triples are stored in a hash table so that the same triple will only appear once and be expanded once. We can further cut the number of expansion by observing that if there are two triples (P, K, C) and (P, K, C') with the same peer set and scan level, except that $C < C'$, we can safely *prune* the triple (P, K, C') because it is suboptimal. Since expanding each triple takes $O(n)$ time, the total running time is $O(nM)$, where M is the total number of triples expanded. In the worst case, there are $O(2^n)$ potential peer sets for each scan level K, and there are $O(n^2)$ scan levels. Therefore, the worst case runtime complexity is $O(nn^2 2^n) = O(n^3 2^n)$. In practice, however, M only grows in the order of $O(1.037^n)$ as observed in our experiments [11]. The algorithm is formally described in Table IV[1].

[1]Given a peer set P, a scan level K, and the terminal set S, `find_merging_point` (P, K, S) finds and returns the merging point m on the next highest scan level (and a flag which is either `TERMINAL`, `STEINER`, or `INVALID`). The operator "$\oplus$" denotes the *pruned* union

Function RSA/DP(S)

```
/*
 *  Given the set of terminals S, return the cost of the
 *  optimal arborescence using the RSA/DP algorithm.
 */

H ← {(φ, ∞, 0)};        // H = the (current) set of triples;
K ← ∞;                  // K = the scan level;
while K ≠ 0 do
    find B ∈ H such that K(B) is maximized;
    m, flag ← find_merging_point(P(B), K(B), S);
    P ← P(B);
    K ← K(B);
    H ← H − {B};
    if flag = TERMINAL then
        H ← H ⊕ {(P − P≻(m) + {m}, |m|,
                  C(B) + Σ_{p∈P≻(m)} |m ⇝ p|)};
    else if flag = STEINER then
        H ← H ⊕ {(P, |r|, C(B))};
        H ← H ⊕ {(P − {m_N, m_E} + {m}, |m|,
                  C(B) + |m ⇝ m_N| + |m ⇝ m_E|)};
    end if
end while
B ← the only element left in H;    // P(B) = {S_1}, K(B) = 0;
return C(B);
```

TABLE I

THE RSA/BNB ALGORITHM.

V. Generalization to the All-Quadrant MRSA Problem

Córdova and Lee proposed a straight-forward generalization of the RSA heuristic to the All-Quadrant MRSA problem, in which the terminals are not restricted to locate in the first quadrant [6]. In particular, they showed that the RSA heuristic can be trivially generalized if "$\langle\rangle$" and "$\prec$" are properly redefined. Furthermore, they showed that the generalization preserves the runtime complexity and performance ratio of the RSA heuristic.

On the other hand, Rao *et al.* [14] observed that the MRSA problem can be reduced to the first quadrant version by considering the subproblem in which the arborescence is restricted to contain the x-axis only between a and b, $a \leq 0 \leq b$, and the y-axis only between c and d, $c \leq 0 \leq d$. The RSA heuristic is used to determine the best Steiner arborescence in each quadrant satisfying this restriction, and the overall Steiner arborescence is obtained by taking the shortest among all restricted Steiner arborescences in the Manhattan plane. We call such method a *stitching*-based generalization.

While the stitching technique is a less efficient generalization of the RSA heuristic than the first approach, it is more appropriate for the RSA/BnB and RSA/DP algorithms. This is because even though the stitching-based technique would require solving a *quadratic* number of single-quadrant MRSA problems, each problem takes it *exponentially* less time to solve than the time to solve it as a single problem. Moreover, there are several effective techniques to reduce the number of single-quadrant problems solved (see [11] for details). In the following, we will call the stitching-based generalization the RSA/DP/AQ algorithm.

of two sets (union followed by pruning of suboptimal triples). This algorithm only returns the cost of the optimal arborescence. Please refer to [11] for details regarding the actual topology generation.

VI. Experimental Results

We have implemented RSA/BnB and RSA/DP for the single-quadrant MRSA problem, and RSA/DP/AQ for the all-quadrant MRSA problem, all in C++. All experiments were run on an HP/UX 9000/770 with 256 MBytes of memory. For comparison purpose, we have also implemented the two exact algorithms proposed by Ho *et al.* [7], which we refer to as HKMS/exp and HKMS/layer. We also compare our algorithms against the optimal branch-and-bound variant of the A-tree algorithm (Atree/BnB) in [4].

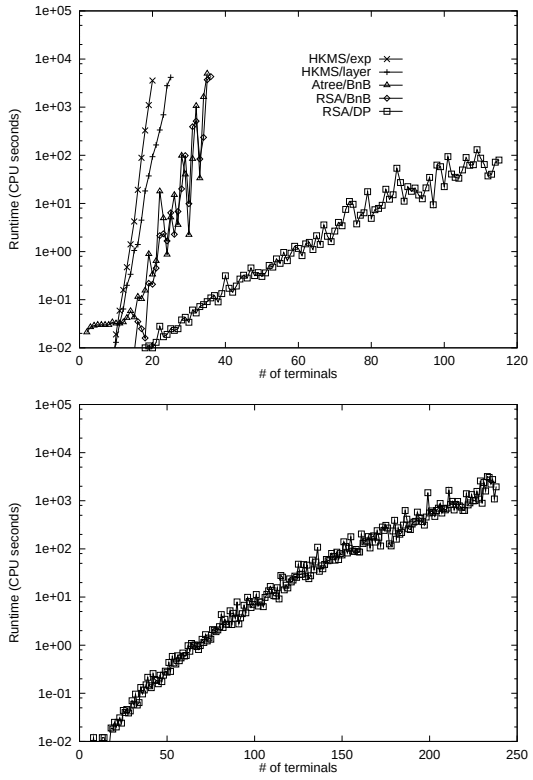

Fig. 1. (a) Average runtime of HKMS/exp, HKMS/layer, Atree/BnB, RSA/BnB, and RSA/DP *vs* number of terminals in the single-quadrant MRSA problem. (b) Average runtime of RSA/DP in the all-quadrant MRSA problem.

A. The Single-Quadrant MRSA Problem

For each integer $1 \leq n \leq 120$, we generate 10 instances of size n with the source S_1 at $(0, 0)$, and the rest of the terminals randomly placed in $[0, 4000]^2$. Figure 1(a) gives the runtimes of the five algorithms (HKMS/exp, HKMS/layer, Atree/BnB, RSA/BnB, and RSA/DP). HKMS/exp and HKMS/layer can handle up to 20 and 25 terminals in one hour, and Atree/BnB and RSA/BnB can handle up to 35 terminals in one hour. RSA/DP, on the other hand, significantly outperforms the four algorithms, and it can handle more than 100 terminals within several minutes. While RSA/DP is based on RSA/BnB, the *irredundancy* achieved by hashing-based dynamic programming results in significantly lower runtime.

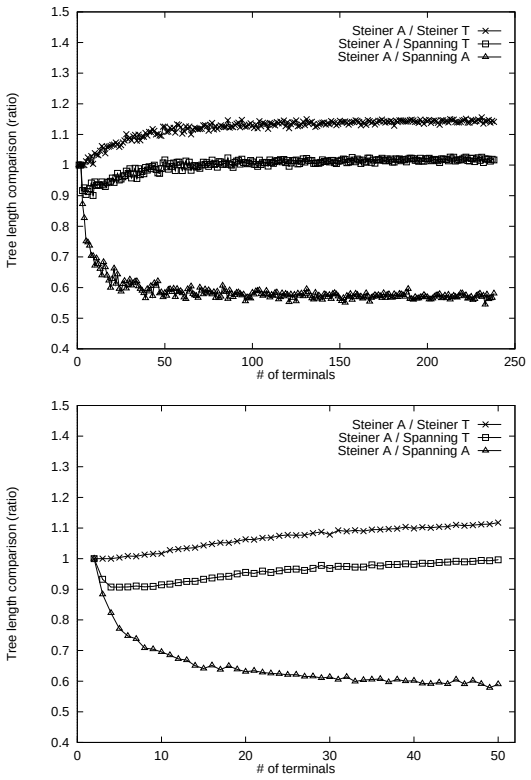

Fig. 2. Length of the MRSA normalized to the corresponding minimum rectilinear spanning arborescence, minimum rectilinear spanning tree, and the rectilinear Steiner tree obtained using the I1S algorithm for (a) terminal sets of size up to 250 (10 instances per size), and (b) terminal sets of size up to 50 (100 instances per size).

B. The All-Quadrant MRSA Problem

For each integer $1 \leq n \leq 250$, we generate 10 instances of size n with the source S_1 at $(0, 0)$, and the rest of the terminals randomly placed in $[-4000, 4000]^2$. Figure 1(b) shows the average runtime of the stitching-based RSA/DP/AQ algorithm. Comparing to RSA/DP, the all-quadrant version can handle a lot more terminals within the same amount of time (and less memory). In fact, RSA/DP/AQ can handle up to 240 terminals in one hour.

C. Comparison with Other Topologies

Figure 2 compares the length of the MRSA obtained by the RSA/DP/AQ algorithm with different topologies including (1) rectilinear Steiner tree (generated by the I1S algorithm [8]), (2) rectilinear spanning tree (optimal), and (3) rectilinear spanning arborescence (optimal). Figure 2(a) shows the average length of the Steiner arborescence normalized to the corresponding Steiner tree, minimum spanning tree, and minimum spanning arborescence, respectively, for each instance. Experimental result shows that the minimum rectilinear Steiner arborescence is consistently about -42.6%, 0.1%, and 12.1% more expensive than the corresponding minimum spanning arborescence, minimum spanning tree, and rectilinear Steiner tree, respectively, for terminal sets of size between 50 and 240.

We also conducted extensive experiments to compare the topologies for small terminal sets. In particular, for $2 \leq n \leq 50$, we generated 100 random instances of size n in $[-4000, 4000]^2$. The length comparison is shown in Figure 2(b). The result clearly shows that rectilinear Steiner arborescences have very small length overhead compared to rectilinear Steiner trees for small number of terminals. In fact, the average difference is less than 1.7% for up to 10 terminals. Furthermore, for four or less terminals, we can prove that a MRSA is also a minimum rectilinear Steiner tree. Therefore, given these observations and the results shown in [3] that rectilinear Steiner arborescence can reduce the interconnect delay by up to 43%, we arrive at the conclusion that rectilinear Steiner arborescence is a very effective topology for interconnect optimization.

VII. Conclusion

In this paper, we have presented RSA/BnB and RSA/DP, two fast optimal algorithms for solving the MRSA problem. We have generalized both algorithms to solve the all-quadrant MRSA problem, and presented extensive experimental results showing the effectiveness of our algorithms.

References

[1] M. J. Alexander, and G. Robins, "New Performance-Driven FPGA Routing Algorithms", *Proc. ACM/IEEE Design Automation Conf.*, 1995, pp. 562 – 567.

[2] H. A. Choi, A. H. Esfahanian, and B. C. Houck, "Optimal Communication Trees with Application to Hypercube Multicomputers", *Proc. Sixth Int'l Conf. on the Theory and Application of Graph Theory*, 1988, pp. 245 – 264.

[3] J. Cong, K. S. Leung, and D. Zhou, "Performance Driven Interconnect Design Based on Distributed RC Delay Model", *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 606 – 611.

[4] J. Cong and K. S. Leung, "On the Construction of Optimal or Near-Optimal Steiner Arborescence", *UCLA Computer Science Tech. Report CSD-960033*, 1996.

[5] J. Cong and P. H. Madden, "Performance Driven Routing with Multiple Sources", *Proc. Int'l Symp. on Circuits and Systems*, 1995, pp. 1157 – 1169.

[6] J. Córdova, Y. H. Lee, "A Heuristic Algorithm for the Rectilinear Steiner Arborescence Problem", *University of Florida CIS Department Tech. Report TR-94-025*, 1994.

[7] J. M. Ho, M. T. Ko, T. H. Ma, and T. Y. Sung, "Algorithms for Rectilinear Optimal Multicast Tree Problem", *Proc. Int'l. Symp. on Algorithms and Computation*, 1994, pp. 106 – 115.

[8] A. B. Kahng and G. Robins, "A New Class of Iterative Steiner Tree Heuristics with Good Performance", On Optimal Interconnections for VLSI", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 11 (1992), pp. 893 – 902.

[9] A. B. Kahng and G. Robins, "On Optimal Interconnections for VLSI", pp. 91 – 95, *Kluwer Academic Publishers*, 1995.

[10] R. R. Ladeira de Matos, "A Rectilinear Arborescence Problem", *Dissertation*, University of Alabama, 1979.

[11] K. S. Leung and J. Cong, "Fast Optimal Algorithms for the Minimum Rectilinear Steiner Arborescence Problem", *UCLA Computer Science Tech. Report CSD-960037*, 1996.

[12] L. Nastansky, S. M. Selkow, and N. F. Stewart, "Cost Minimal Trees in Directed Acyclic Graphs", *Zeitschrift für Operations Research*, 18 (1974), pp. 59 – 67.

[13] T. Okamoto and J. Cong, "Interconnect Layout Optimization by Simultaneous Steiner Tree Construction and Buffer Insertion", *Proc. IEEE Int'l Conf. on Computer-Aided Design*, 1996, pp. 44 – 49.

[14] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor, "The Rectilinear Steiner Arborescence Problem", *Algorithmica*, 7 (1992), pp. 277 – 288.

[15] G. E. Téllez and M. Sarrafzadeh, "On Rectilinear Distance-Preserving Trees" *Proc. IEEE Symp. on Circuits and Systems*, 1 (1995), pp. 163 – 166.