A Study of Optimal Cost-Skew Tradeoff and Remaining Suboptimality in Interconnect Tree Constructions

Kwangsoo Han[‡], Andrew B. Kahng[†‡], Christopher Moyes[†] and Alex Zelikovsky[*]

[†]CSE and [‡]ECE Departments, UC San Diego, La Jolla, CA, USA
[*]Department of Computer Science, Georgia State University, Atlanta, GA, USA
{kwhan, abk, cmoyes}@ucsd.edu, alexz@cs.gsu.edu

ABSTRACT

Cost and skew are among the most fundamental objectives for interconnect tree synthesis. The cost-skew tradeoff is particularly important in buffered clock tree construction, where clock subnets are an important "sweet spot" for balancing on-chip variation-aware analysis, skew, power and other factors. In advanced nodes, where both performance and power are critical to IC products, there is a renewed challenge of minimizing wirelength while controlling skew. In this work, we formulate the minimum-cost bounded skew spanning and Steiner tree problems as flow-based integer linear programs, and give the first-ever study of optimal cost-skew tradeoffs. We also assess heuristics (notably, Bounded-Skew DME (BST-DME), Steiner shallow-light tree (SALT), and Prim-Dijkstra (PD)) that are currently available for trading off cost and skew. Experimental results demonstrate that BST-DME has suboptimality $\sim$ 10% in cost at iso-skew and $\sim$ 50% in skew at iso-cost. In addition, SALT and PD shows suboptimality in terms of skew by up to $\sim 3\times$.

1 INTRODUCTION

The difficulty of scaling integrated-circuit power efficiency, performance, area and cost (PPAC) in advanced technology nodes has been well-documented [46]. With the lack of new back-end-of-line interconnect materials, and consequent poor scaling of wire resistance and capacitance, there is increased pressure to improve the quality of interconnect layout. The recent paper of Alpert et al. [1] notes the power-sensitivity of modern (mobile, IoT, etc.) designs: "a 1% reduction in power is viewed as a big win for ... physical implementation", and "even a small WL savings with similar timing can have a high impact on value". In other words, there is renewed focus on the *cost* of interconnect trees in advanced VLSI.

Clock distribution has long been a crucial aspect of IC physical implementation since it strongly affects both power and performance. Clock routing brings together both cost and *skew* criteria: the cost-skew tradeoff [11] is particularly important in buffered clock tree construction, where clock subnets with ~20 fanouts are a "sweet spot" for balancing of on-chip variation-aware analysis, skew, power and other factors. Future growth in the number of fanouts per clock buffer is unlikely, as fanout is limited by poor scaling of drive strengths relative to interconnect parasitics, increased use of multi-bit flip-flops to reduce clock wirelength and power, and increasing number of clocks in complex, low-power SOCs.[1]

Over the past decades, a number of works have studied the *bounded-skew routing tree* problem:

Bounded-Skew Routing Tree (BST) Problem. Given a set of terminals (points in the Manhattan plane) $P = \{p_1, p_2, \ldots, p_n\}$ with p_1 being the designated *root* (source) terminal, along with a skew bound B, construct a tree T with minimum cost $c(T)$ that contains all points of P and with root-terminal pathlength skew $\leq B$, i.e., $|d_T(1,i) - d_T(1,j)| \leq B$, for $2 \leq i < j \leq n$.

Here, the cost of an edge in T is its Manhattan length. The cost $c(T)$ of the tree T is the sum of its edge costs, and $d_T(1,i)$ denotes the sum of edge costs along the unique path in T from p_1 to p_i. The BST problem may be formulated in either the *spanning* or the *Steiner* contexts; we denote these respectively as the **BSSpanT** and the **BSSteinT** problems. Figure 1 illustrates the BST problem as well as the different nature of the BSSpanT and BSSteinT problems. Figure 1(a) shows the distribution of terminals for an example with $n = 8$. Figure 1(b) shows cost-skew tradeoff for this instance. Note that when $B = \infty$, the BSSpanT problem is the same as the rectilinear minimum spanning tree problem, and the BSSteinT problem is the same as the rectilinear Steiner minimum tree problem. Further, when $B = 0$ the BSSteinT problem becomes the exact zero-skew clock tree (ZST) problem studied in [5] [7] and many subsequent works. The unknown that we study in this work is the tradeoff between these extreme solutions.

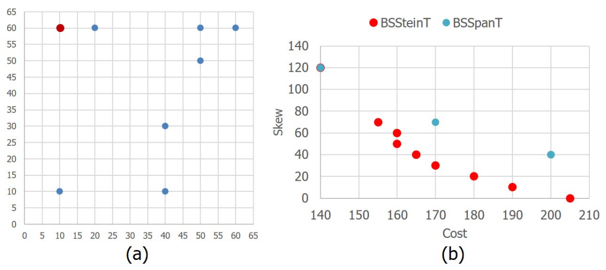


Figure 1: Illustration of the bounded-skew spanning and Steiner tree problems. (a) Distribution of terminals for an example with $n = 8$. (b) Minimum achievable BSSteinT cost decreases as the skew bound B increases. For the same instance and the same values of B, a BSSpanT may not always exist.

The VLSI CAD literature of the 1990s developed constructions for bounded-skew clock and Steiner routing trees [11] [20] [24].

[1]A recent keynote address of TI's Anthony Hill [18] cites an IOT design with 200k instances, 200 distinct source clocks, an average of 12 clocks per register, and 1200 total clock domains.

SLIP '18, June 23, 2018, San Francisco, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5900-9/18/06…$15.00
https://doi.org/10.1145/3225209.3225215

Additionally, cost-radius tradeoff methods such as shallow-light trees [12] [26] (see also [13] [25] [15] [16]) or Prim-Dijkstra trees [1] [2] were applied in the bounded-skew context, since bounding the radius of a tree trivially also bounds the skew of a tree. During the 2000s, the discrete algorithms community addressed bounded-skew tree construction in works such as [8] [39]. In recent advanced nodes, where both performance and power are critical to IC products, there is now intense focus on the challenge of minimizing clock distribution wirelength while controlling skew.

Our present work revisits the bounded-skew routing tree problem – in both the spanning tree and Steiner tree contexts – with an aim to determine "how much is left on the table". While VLSI interconnect trees are ultimately realized as Steiner trees, the spanning formulation is of distinct interest. Footnote 1 of [1] observes that "For global routing, spanning trees are often preferred to Steiner trees since global routing commonly decomposes multi-fanout nets into two-pin nets. A spanning tree provides the router with an obvious decomposition. However, Steiner trees are not well-suited for this because the Steiner points become unnecessary constraints that restrict the freedom of the router to resolve congestion." Indeed, the spanning and Steiner formulations have different "behaviors": Elkin and Solomon [16] show that Steiner shallow-light trees can be exponentially lighter than their spanning counterparts (recall also Figure 1).

The key contributions of this work are as follows.

- We formulate the minimum-cost bounded skew spanning and Steiner tree problems as flow-based integer linear programs and give the first-ever study of *optimal* cost-skew tradeoffs.
- We evaluate the heuristics (Bounded-Skew DME and Prim-Dijkstra variants) that are currently the best available methods for trading off cost and skew, and quantify remaining suboptimality.
- In the Appendix, we apply dynamic programming to obtain insight into optimal solutions of a highly restricted *one-dimensional* (1-D) BST problem; this enables us to shed some light on such "classic" questions as optimal skew budgeting across levels of a tree.

In the following, Section 2 summarizes related work on cost-delay tradeoffs and bounded-skew tree constructions. Section 3 describes flow-based ILP formulations for the spanning and Steiner BST problems. Section 4 experimentally demonstrates a surprisingly substantial "gap" between existing heuristic tree constructions and optimum bounded-skew trees. We conclude in Section 5 with ongoing and future directions. In the Appendix, we analyze the nature of optimal dynamic programming-based solutions to 1-D instances of the BST problem, and present observations regarding "skew budgeting" and the impact of clustering in sink placement.

2 RELATED WORK

Many spanning and Steiner tree heuristics have been proposed for VLSI routing applications. These heuristics typically optimize or trade off the fundamental objectives of tree cost, delay and skew [22]. Types of tree constructions for VLSI that are related to our present work can be classified into three main categories: cost-delay tradeoffs, bounded-skew constructions, and optimal (integer programming-based) constructions.

Cost-delay tradeoffs have most famously been achieved by *shallow-light* constructions, which optimize cost (wirelength) and radius (maximum source-sink pathlength) simultaneously to within constant factors of optimal. For example, the BRBC algorithm [12] produces a tree that has wirelength no greater than $1 + 2/\epsilon$ times that of a minimum spanning tree (MST), and radius no greater than $1 + \epsilon$ times that of a shortest-paths tree (SPT). Over the ensuing 25+ years, numerous works ranging from [26] to [16] [9] have continued to improve the basic approach. The SALT method of [9] is the most recent and strongest work in the shallow-light literature, incorporating additional techniques such as post-processing via edge flipping [19]. The *Prim-Djkstra* algorithm [2] achieves in practice a high-quality tradeoff between tree cost and maximum source-terminal pathlength (i.e., radius), but has no provable shallow-light property. The very recent work of [1] improves the original Prim-Dijkstra method with topology and edge-flipping optimizations, and can produce tree solutions superior to [9]. Additional work has studied the *rectilinear Steiner arborescence* (RSA) problem, which seeks to find a minimum-cost tree that achieves optimal source-sink delay *at every sink*, i.e., a minimum-cost shortest-paths Steiner tree. Rao et al. [32] and Cong et al.[14] give heuristics for the RSA problem, which is known to be NP-complete [35]; an implementation of the A-Tree method of [14] is available at [44].

Bounded-skew tree (BST) constructions originally arose as extensions of deferred-merge embedding (DME) based zero-skew tree (ZST) constructions [5] [7] [23]. Notably, [11] [20] [24] all extend the DME algorithm to achieve BST routing. With a skew bound of $B = 0$, the BST problem reduces to the ZST problem. When $B = \infty$, the BST problem reduces to the rectilinear Steiner minimum tree (RSMT) problem. Tsao and Koh [37] improve the DME algorithm's bottom-up merging step to construct trees subject to general skew constraints. Empirical results show improvements over BST-DME with certain skew constraints. In the discrete algorithms literature, works of Charikar et al. [8] and of Zelikovsky and Mandoiu [39] propose ZST and BST heuristics with constant-factor error bounds; the latter work gives a realizable ZST construction based on the "rooted-Kruskal" approach which guarantees rectilinear BST cost within 9 times of optimal. Rajaram et al. [31] apply bounded-skew tree construction within low-cost (crosslink insertion-based) non-tree routing. Below, we study cost-skew tradeoff performance of an updated version [36] of the open-source BST-DME implementation of Tsao [45].

A number of *optimal* tree constructions have been proposed as well. The well-known FLUTE method of Chu and Wong [10] uses topology pruning and look-up tables to find RSMT solutions extremely efficiently; FLUTE solutions are optimal for instances with up to ~9 pins. The well-known GeoSteiner code of Warme et al. [43] can solve the RSMT problem optimally for instances with thousands of points. The work of Peyer et al. [30] exemplifies the use of integer linear programming (ILP) to solve the Steiner tree problem via the flow-based *directed Steiner tree* framework. Single-commodity and multi-commodity flow-based ILPs have been also used by Han et al. [17] to assess back-end-of-line design rule impacts on local routability, and by Jia et al. [21] within a detailed router. Aneja [4] applies a set-covering ILP to the construction of Steiner trees given a prescribed set of Steiner points. A "row generation" technique prevents an exponential number of constraints from arising. Oh et al. [29] use linear programming to find Steiner routing trees with upper-and lower-bounded path delays within a prescribed topology; a method similar to BST-DME is used to embed the Steiner points in the Manhattan plane. No previous work that we are aware of optimally solves either the spanning or the Steiner form of the bounded-skew tree problem. Below, we experimentally study a flow-based ILP that solves both the spanning and Steiner BST formulations.

3 FLOW-BASED ILP FORMULATION

We now formulate an ILP that can be generally applied to both the BSSpanT and BSSteinT problems. In this section, we first introduce our bounded-skew tree routing formulation. Second, we then explain constraints that detect and block any cycles, such that the ILP outputs a well-formed tree as its solution. Third, additional constraints to improve runtime are explained. Table 1 lists the notations that we use.

Table 1: Notations

Notation	Meaning
p_i	i^{th} point ($p_i \in P$, p_1 is a root point)
v_i	i^{th} vertex ($v_i \in V$, $P \subseteq V$)
e_{jk}	a directed edge from vertex v_j to vertex v_k ($e_{jk} \in E$)
λ_{jk}	0-1 indicator of whether e_{jk} is in a tree T
c_{jk}	cost of edge e_{jk}
f^i_{jk}	0-1 indicator of whether the flow to p_i goes through e_{jk}
d^i_j	pathlength at v_j along unique v_1-p_i path
m_{ij}	Manhattan distance from v_i to v_j
B	skew bound
L	lower bound on pathlength from source v_1

Minimize: $\sum_{j,k} \lambda_{jk} \cdot c_{jk}$

Subject to:

$$\lambda_{jk} \geq f^i_{jk} \; \forall p_i \in P,\; e_{jk} \in E \tag{1}$$

$$\sum_j f^i_{jk} - \sum_j f^i_{kj} = \begin{cases} 1 \text{ if } v_k = v_1, \forall v_j \in V,\; p_i \in P,\; i \neq 1 \\ -1 \text{ else if } v_k = v_i \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

$$\sum_{j,k} c_{jk} \cdot f^i_{jk} \geq L \; \forall p_i \in P,\; i \neq 1 \tag{3}$$

$$\sum_{j,k} c_{jk} \cdot f^i_{jk} \leq L + B \; \forall p_i \in P,\; i \neq 1 \tag{4}$$

3.1 Bounded-Skew Tree Routing

Given a set of terminals P, specified as (x, y) points in the Manhattan plane with x, y integers, we create a graph $G = (V, E)$ whose vertex set V contains P as well as additional points S (i.e., $V = P \cup S$). (Thus, each point in P is identified with some vertex in V.) For the BSSpanT problem, S is empty and E consists of all $|P| \cdot (|P| - 1)$ possible directed edges between pairs of terminals. Thus, $G = (V, E)$ is a complete graph in the BSSpanT problem. For the BSSteinT problem, S is a set of non-terminal points in a *half-integer grid* of P, and E is a set of directed edges between any neighboring vertices. Each vertex has up to four outgoing and four incoming edges to/from neighbor vertices in the east, west, north and south directions. The half-integer grid consists of all points in the bounding box of P for which both x and y coordinates are multiples of $1/2$. By convention, we assume that (i) $v_1 = p_1$ is the *root* (i.e., source) terminal, (ii) $\{v_2, v_3, ..., v_{|P|}\} = \{p_2, p_3, ..., p_{|P|}\}$ are the *leaf* (i.e., sink) terminals, and (iii) other vertices $\{v_{|P|+1}, ...v_{|V|}\}$ are additional non-terminal vertices. We formulate the following integer linear program:

Our objective is to minimize total cost, while satisfying a given skew bound B. We consider each path from p_1 to p_i as a separate flow. λ_{jk} is a global binary variable that indicates whether any flow goes through an edge e_{jk} in the tree solution T. Constraint (1) forces $\lambda_{jk} = 1$ when a flow exists in e_{jk}.

Flow conservation. Constraints (2) are for flow conservation. (A unit of flow from source v_1 to sink v_i will traverse a path from p_1 to p_i.) These constraints enforce that (i) there is one net outgoing unit of flow at a vertex v_k that is the vertex identified with the root terminal p_1; (ii) there is one net incoming unit of flow at a vertex v_k that is the vertex identified with the leaf terminal p_i, and (iii) otherwise, the sum of incoming and outgoing flow at vertex v_k must be zero. Since each flow for each path is considered exclusively, for the flow to the terminal p_i, other terminals' vertices (i.e., $v_2, ..., v_{|P|}$ except for v_i) are not considered as leaf terminals.

Skew bound constraints. Given L and B, Constraints (3) - (4) respectively bound the minimum and maximum pathlengths for all source-to-sink paths. However, with these constraints only, invalid solutions that contain cycles could arise. Next, we add more constraints to block the formation of cycles.

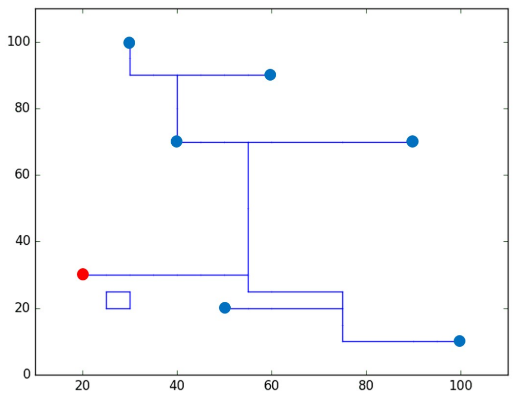


Figure 2: Example solution with a cycle on the lower-left corner. Red dot is a root and blue dots are leaf terminals.

3.2 Cycle Correction

Figure 2 shows an example solution with a cycle. This solution satisfies the above flow conservation constraints since the sum of incoming and outgoing flows are the same for each vertex in the cycle. This happens when the cost of creating the cycle is less than the cost of detouring to leaf terminals (that are close to the root) in order to satisfy a given skew bound. To prevent the cycle, we define a new d^i_j variable that represents the pathlength from p_1 to v_j for path p_i. The d^i_j should satisfy the following constraints (5):

$$\begin{cases} d^i_j = 0 \;\text{ if } v_j = v_1, \forall\, p_i \in P,\; i \neq 1 \\ d^i_j \geq L \text{ else if } v_j = v_i, \forall\, p_i \in P,\; i \neq 1 \\ d^i_j \leq L + B \text{ otherwise } \forall\, p_i \in P,\; i \neq 1 \end{cases} \tag{5}$$

In other words, the pathlength from the root to any vertex v_j must lie between prescribed minimum and maximum pathlength bounds.

To compute d_j^i, we add the following constraints.

$$d_j^i + U \cdot (1 - f_{kj}^i) \geq d_k^i + c_{kj} \; \forall v_j, v_k \in V, j \neq k, \; p_i \in P \quad (6)$$

$$d_j^i - U \cdot (1 - f_{kj}^i) \leq d_k^i + c_{kj} \; \forall v_j, v_k \in V, j \neq k, \; p_i \in P \quad (7)$$

When $f_{kj}^i = 1$, d_j^i should be equal to d_k^i. Otherwise, these constraints are always met with a large constant value U. With these constraints, d_j^i becomes infinite if there is a cycle.

3.3 Constraints for Runtime Improvement

It is well known that the chief drawback of using ILP is long, poorly-scaling runtime. We add further constraints to predetermine some variables according to what we might know before we run the ILP instance. The idea is to find flow variables that cannot exist, or combinations of variables that cannot coexist. This reduces the solution space for an ILP solver to explore.

$$\text{if } m_{1j} + m_{ji} > L + B \; \forall v_j \in V, \; j \neq 1, ..., |P|, \; p_i \in P$$
$$f_{jk}^i = 0, \; f_{kj}^i = 0 \; \forall e_{jk} \in E, \; e_{kj} \in E \quad (8)$$

For any non-terminal vertex v_j, if the sum of Manhattan length from root terminal p_1 to v_j and from v_j to leaf terminal p_i is larger than the upper bound on pathlength L+B, all incoming and outgoing flows for p_i going through v_j should be zero.

Similarly, we can consider two non-terminal vertices v_j and $v_{j'}$:

$$\text{if } m_{1j} + m_{jj'} + m_{j'i} > L + B$$
$$\&\& \; m_{1j'} + m_{j'j} + m_{ji} > L + B$$
$$f_{jk}^i + f_{j'k'}^i = 1, \; f_{kj}^i + f_{j'k'}^i = 1, \; \forall e_{jk} \in E, \; e_{kj} \in E \quad (9)$$
$$f_{jk}^i + f_{k'j'}^i = 1, \; f_{kj}^i + f_{k'j'}^i = 1, \; \forall e_{j'k'} \in E, \; e_{k'j'} \in E \quad (10)$$

For any two non-terminal vertices v_j and $v_{j'}$, if the sum of Manhattan lengths from p_1 going through v_j and $v_{j'}$ to terminal p_i is larger than the pathlength upper bound, then any combinations of incoming and outgoing flows going through v_j and $v_{j'}$ cannot coexist. Our ILP implementation applies such additional constraints to reduce ILP solver runtime.

3.4 Analysis of the Number of Variables and Constraints

The number of variables and constraints depends on the number of edges ($|E|$), vertices ($|V|$) and points ($|P|$).

- The number of variables λ_{jk} is $|E|$.
- The number of variables f_{jk}^i is $|E| \cdot |P|$.
- The number of variables d_j^i is $|V| \cdot |P|$.
- The number of Constraints (1) is $|E| \cdot |P|$.
- The numbers of Constraints (2), (5), (6), (7) are $|V| \cdot |P|$.
- The numbers of Constraints (3) and (4) are (each) $|P|$.

4 EXPERIMENTAL SETUP AND RESULTS

4.1 Experimental Setup

We implement our tool in C++ and use CPLEX 12.6.1[47] as our ILP solver. The following experiments are performed on a 2.7 GHz Intel Xeon server with 32 threads.

Even with the additional constraints for runtime improvement, our flow-based ILP for Steiner tree only works for a limited condition (i.e., $|P| \leq \sim 16$, $|V| \leq \sim 140$, $|E| \leq \sim 550$). Under this condition, we generate 50 testcases for each $|P| = \{8, 10, 12, 14, 16\}$. The terminals of each testcase are randomly distributed.

For the generated testcases, we run our flow-based ILP for Steiner tree, and obtain cost-skew tradeoff curves. For each instance, we run ILP with four different skew bounds $= M \cdot \{0.3, \; 0.5, \; 0.7, \; \infty\}$, where M is the maximum source-to-sink Manhattan length. We do not run our ILP with skew bound $B = 0$ due to the long runtime. We also set $L = M - B$ for a given B. When a fixed L is used, our ILP Steiner tree solution could end up with a suboptimal solution. The impact of a fixed L on suboptimality is discussed in Section 4.2.

We also run several academic tools for evaluation; BST-DME [45], SALT [9] and PD [2]. For each tool, we sweep input parameters to obtain several solutions.

4.2 Experimental Results

Study of cost-skew tradeoff. We normalize the costs (resp. skews) of academic tools' solutions as well as our ILP-based spanning tree solutions by the minimum cost (resp. skew) achieved from ILP-based Steiner tree for each testcase. Figure 3(a) shows the results for one 14-terminal instance. Each data point is mapped to a solution from the corresponding tool. This figure clearly shows that our ILP-based Steiner tree solutions are dominating the other tools' solutions on both skew and cost. Some solutions from BST-DME achieve slightly better skew than our minimum skew from ILP-based Steiner tree solutions. This is because we do not have a run with $B = 0$. Figure 4 shows the plots of ILP-based Steiner tree solutions for this 14-terminal instance.

For a more comprehensive study across different instances, we propose the following way for comparison. (1) For each tool, we select three representative solutions: the minimum-skew solution, the minimum-cost solution, and a "median" solution in between. (2) We then compute the average normalized cost (resp. skew) for the same set of solutions (e.g., minimum skew solutions) for all 50 instances.

Figures 3(b)-(f) show the cost-skew tradeoff curves for all $|P| = \{8, 10, 12, 14, 16\}$, respectively. From these figures, we observe that:

- ILP-based Steiner tree dominates all other tools in terms of both cost and skew across all terminal nets.
- Compared to the Steiner tradeoff curve, BST-DME is $\sim$ 10% suboptimal in cost at iso-skew and $\sim$ 50% suboptimal in skew at iso-cost. Solutions for different terminal nets show a similar trend.
- Both SALT and PD mostly generate large skew solutions, with up to $\sim 3\times$ suboptimality. This suggests that optimizing shallowness only, without awareness of skew, is insufficient to find solutions with good skew.
- In general, ILP-based spanning tree solutions have larger skew and cost than the ones from SALT and BST-DME. However, as the number of terminals increases, it generates comparable or better cost and skew solutions than SALT and BST-DME. This implies that the optimal spanning tree solutions could be good candidates for high-fanout nets.

Runtime. ILP for spanning tree runs very fast. Average runtime is 0.32 second and maximum runtime is 4 second. On the other hand, ILP for Steiner tree uses the half-integer grid to ensure that optimal solutions. Thus, as we increase the number of terminals, the number of vertices and edges increase and runtime goes up quickly. Runtime also increases as the skew bound is tightened. Table 2 shows the average runtime for different $|P|$ and skew bound.

Possible suboptimality with fixed L. Due to runtime scaling and available computational resource, we have used a fixed L (computed

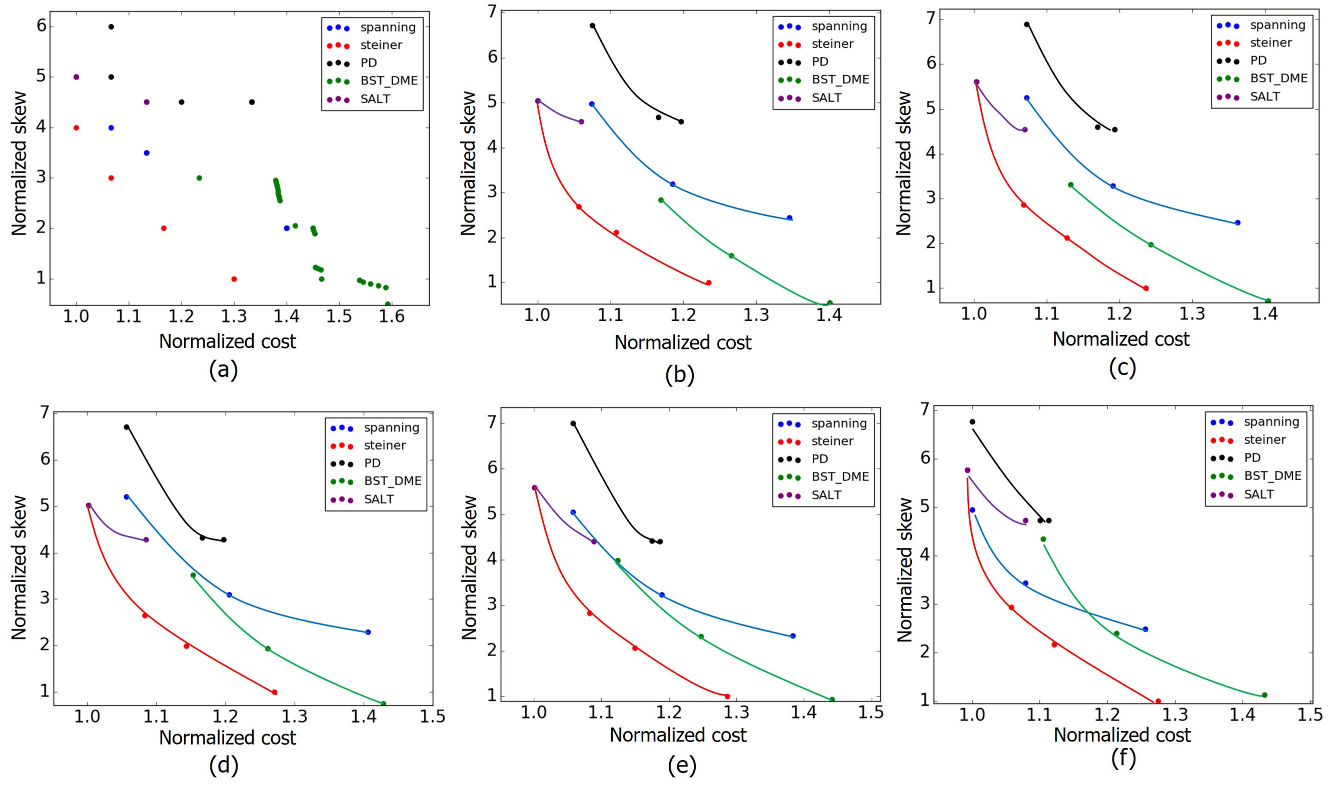


Figure 3: Illustration of cost-skew tradeoff. (a) cost-skew tradeoff curve for one 14-terminal instance (b) cost-skew tradeoff curves for all 8-terminal instances (c) cost-skew tradeoff curves for all 10-terminal instances (d) cost-skew tradeoff curves for all 12-terminal instances (e) cost-skew tradeoff curves for all 14-terminal instances (f) cost-skew tradeoff curves for all 16-terminal instances.

Table 2: Average ILP runtime for Steiner tree.

Skew bound	$\|P\| = 8$	$\|P\| = 10$	$\|P\| = 12$	$\|P\| = 14$	$\|P\| = 16$
Unbounded	0.33	0.50	0.68	1.04	0.78
$0.7 \cdot M$	2.89	5.22	41.24	130.08	58.53
$0.5 \cdot M$	13.20	74.98	364.03	1522.25	892.50
$0.3 \cdot M$	320.77	662.01	2593.59	4664.06	3477.73

as $M - B$; see Section 4.1 above) in our reported results for ILP-based bounded-skew Steiner trees. However, a fixed L can cause small suboptimality in the solutions. To study the impact of L on suboptimality, we select 10 sample instances (two instances from each testset with a given number of terminals) and vary L from $M - B$ to M. We then compare the best cost found to the cost obtained using a fixed $L = M - B$. We find that one out of 10 nets is suboptimal due to the fixed L and the suboptimality is 2.8%.

5 CONCLUSIONS

In this work, we empirically study the minimum-cost bounded skew spanning and Steiner tree problems. We formulate and apply a flow-based ILP to find optimal cost-skew tradeoffs for generated testcases with number of terminals from 8 to 16. Based on the optimal cost-skew tradeoffs, we find significant remaining suboptimality of several state-of-art academic tools: (1) BST-DME, (2) SALT, and (3) Prim-Dijkstra. Across our testcases, BST-DME has suboptimality $\sim$ 10% in cost at iso-skew, and $\sim$ 50% in skew at iso-cost. In addition, SALT and PD show suboptimality in terms of skew by up to $\sim 3\times$. This degree of suboptimality is very different from the near-optimality in practice of heuristics for the RSMT problem (e.g., FLUTE, 1-Steiner, etc.). Thus, our study motivates renewed attention to the cost-skew tradeoff.

Our future work includes (1) further scalability study and improvement of the flow-based ILP formulation; (2) extensions of our suboptimality study to the cost-radius tradeoff and well-studied variants (sink-specific radius bounds, critical-sink trees, required arrival time (RAT) trees, etc.), and (3) benchmark suite generation with known optimal solutions to various formulations.

ACKNOWLEDGMENTS

A. Zelikovsky's research is supported in part by NSF under grant number CCF-1619110. Research in UCSD ABKGroup is supported in part by funding from NSF, Qualcomm, Samsung, NXP, Mentor Graphics and the C-DEN center.

REFERENCES

[1] C. J. Alpert, W.-K. Chow, K. Han, A. B. Kahng, Z. Li, D. Liu and S. Venkatesh, "Prim-Dijkstra Revisited: Achieving Superior Timing-driven Routing Trees" *Proc. ISPD*, 2018, pp. 10-17.
[2] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng and D. Karger, "Prim-Dijkstra Tradeoffs for Improved Performance-driven Routing Tree Design", *IEEE Trans. on CAD* 14(7) (1995), pp. 890-896.
[3] C. J. Alpert, A. B. Kahng, C. N. Sze and Q. Wang, "Timing-driven Steiner Trees are (Practically) Free", *Proc. DAC*, 2006, pp. 389-392.
[4] Y. P. Aneja, "An Integer Linear Programming Approach to the Steiner Problem in Graphs", Networks 10(2), 1980, pp. 167-178.
[5] K. Boese and A. B. Kahng, "Zero-Skew Clock Routing Trees With Minimum Wirelength", *Proc. IEEE ASIC Conf.*, 1992, pp. 1.1.1 - 1.1.5.
[6] M. Borah, R. M. Owens and M. J. Irwin, "An Edge-based Heuristic for Steiner Routing", *IEEE Trans. on CAD* 13(12) (1994), pp. 1563-1568.

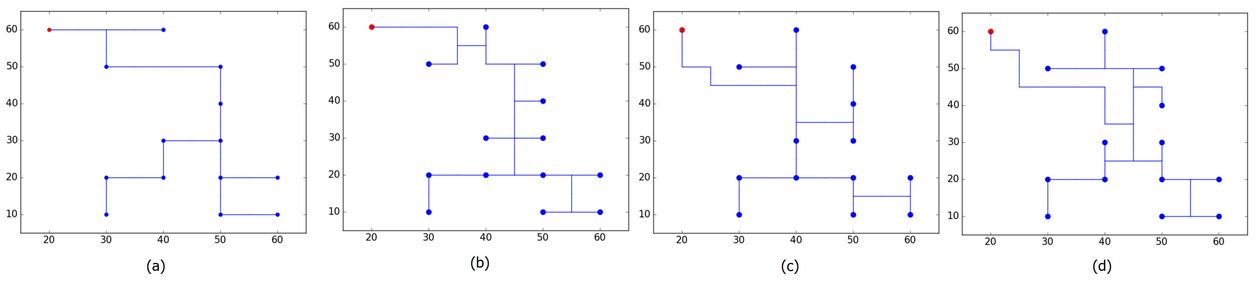


Figure 4: Illustration of ILP-based Steiner tree solutions for a 14-terminal instance with (a) $B = \inf$, (b) $B = 0.7 \cdot M$, (b) $B = 0.5 \cdot M$, (d) $B = 0.3 \cdot M$.

[7] T. H. Chao, Y. C. Hsu, J. M. Ho, K. D. Boese and A. B. Kahng, "Zero Skew Clock Routing With Minimum Wirelength", *IEEE Trans. on Circuits and Systems* 39(11) (1992), pp. 799-814.

[8] M. Charikar, J. Kleinberg, R. Kumar, S. Rajagopalan, A. Sahai and A. Tomkins, "Minimizing Wirelength in Zero and Bounded Skew Clock Trees", *SIAM J. Disc. Math.* 17(4) (2004), pp. 582-595.

[9] G. Chen, P. Tu and E. F. Y. Young, "SALT: Provably Good Routing Topology by a Novel Steiner Shallow-Light Tree Algorithm", *Proc. ICCAD*, 2017, pp. 569-576.

[10] C. Chu and Y.C. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design", *IEEE Trans. on CAD* 27(1) (2008), pp.70-83.

[11] J. Cong, A. B. Kahng, C. K. Koh and C.-W. A. Tsao, "Bounded-Skew Clock and Steiner Routing", *ACM TODAES* 3(3) (1998), pp. 341-388.

[12] J. Cong, A. B. Kahng, G. Robins and M. Sarrafzadeh, "Provably Good Performance-driven Global Routing", *IEEE Trans. on CAD* 11(6) (1992), pp. 739-752.

[13] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh and C.K. Wong, "Performance-driven Global Routing for Cell Based ICs", *Proc. ICCD*, 1991, pp. 170-173.

[14] J. Cong, K. S. Leung and D. Zhou, "Performance-driven Interconnect Design Based on Distributed RC Delay Model", *Proc. DAC*, 1993, pp. 606-611.

[15] M. Elkin and S. Solomon, "Narrow-shallow-low-light Trees With and Without Steiner Points", *SIAM J. Disc. Math.* 25(1) (2011), pp. 181-210.

[16] M. Elkin and S. Solomon, "Steiner Shallow-light Trees are Exponentially Lighter than Spanning Ones", *SIAM J. Comp.* 44(4) (2015), pp. 996-1025.

[17] K. Han, A. B. Kahng and H. Lee, "Evaluation of BEOL Design Rule Impacts Using an Optimal ILP-based Detailed Router", *Proc. DAC*, 2015, pp. 1-6.

[18] A. Hill, keynote address, *Proc. ISPD*, 2018. http://www.ispd.cc/slides/2018/k1.pdf (Slide 17).

[19] J. M. Ho, G. Vijayan and C. K. Wong, "New Algorithms for the Rectilinear Steiner Tree Problem", *IEEE Trans. on CAD* 9(2) (1990), pp. 185-193.

[20] J.-H. Huang, A. B. Kahng and C.-W. A Tsao, "On the Bounded-Skew Clock and Steiner Routing Problems", *Proc. DAC*, 1995, pp. 508-513.

[21] X. Jia, Y. Cai, Q. Zhou, G. Chen, Z. Li and Z. Li, "MCFRoute: A Detailed Router Based on Multi-Commodity Flow Method", *Proc. ICCAD*, 2014, pp. 397-404.

[22] A. B. Kahng and G. Robins, *On Optimal Interconnects for VLSI*, Kluwer Academic Publishers, 1995.

[23] A. B. Kahng and C.-W. A. Tsao, "Planar-DME: A Single-Layer Zero-Skew Clock Tree Router", *IEEE Trans. on CAD* 15(1) (1996), pp. 8-19.

[24] A. B. Kahng and C.-W. A. Tsao, "Practical Bounded-Skew Clock Routing", *J. VLSI Signal Processing* 16 (1997), pp. 199-215.

[25] G. Kortsarz and D. Peleg, "Approximating Shallow-light Trees", *Proc. SODA*, 1997, pp. 103-110.

[26] S. Khuller, B. Raghavachari and N. Young, "Balancing Minimum Spanning Trees and Shortest-path Trees", *Proc. SODA*, 1993, pp. 243-250.

[27] K.-S. Leung and J. Cong, "Fast Optimal Algorithms for the Minimum Rectilinear Steiner Arborescence Problem", *Proc. ISCAS*, 1997, pp. 1568-1571.

[28] A. Lim, S.-W. Cheng and C.-T. Wu, "Performance Oriented Rectilinear Steiner Trees", *Proc. DAC*, 1993, pp. 171-175.

[29] J. Oh, I. Pyo and M. Pedram, "Constructing Lower and Upper Bounded Delay Routing Trees Using Linear Programming", *Proc. DAC*, 1996, pp. 401-404.

[30] S. Peyer, M. Zachariasen and D. G. Jorgensen, "Delay-related Secondary Objectives for Rectilinear Steiner Minimum Trees", *Discrete Appl. Math.* 136(2-3) (2004), pp. 271-298.

[31] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing clock skew variability via crosslinks", *IEEE TCAD*, 25(6) 2006, pp. 1176-1182.

[32] S. K. Rao, P. Sadayappan, F. K. Hwang and P. W. Shor, "The Rectilinear Steiner Arborescence Problem", *Algorithmica* 7(2) (1992), pp. 277-88.

[33] L. Scheffer, Bookshelf RMST code, http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/RMST.

[34] R. Scheifele, "Steiner Trees with Bounded RC-delay", *Algorithmica* 78(1) (2017), pp. 86-109.

[35] W. Shi and C. Su, "The Rectilinear Steiner Arborescence Problem is NP-complete", *SIAM J. Comp.* 35(3) (2006), pp. 729-740.

[36] B. Taskin, *personal communication*, Oct. 2012.

[37] C.-W. A. Tsao and C.-K. Koh. "UST/DME: A Clock Tree Router for General Skew Constraints", *ACM Trans. on Design Automation of Electronic Systems* 7(3), 2002, 359-379.

[38] D. M. Warme, P. Winter and M. Zachariasen, "Exact Algorithms for Plane Steiner Tree Problems: A Computational Study", in D.Z. Du, J.M. Smith and J.H. Rubinstein (Eds.) *Advances in Steiner Trees*, Kluwer Academic Publishers, 2000, pp. 81-116.

[39] A. Z. Zelikovsky and I. I. Mandoiu, "Practical Approximation Algorithms for Zero- and Bounded-skew Trees", *SIAM J. Disc. Math.* 15(1) (2002), pp. 97-111.

[40] C. J. Alpert, R. G. Gandham, J. Hu, S. T. Quay and A. J. Sullivan, "Apparatus and Method for Determining Buffered Steiner Trees for Complex Circuits", *U.S. Patent 6,591,411*, July 2003.

[41] G. M. Furnish, M. J. LeBrun and S. Bose, "Node Spreading Via Artificial Density Enhancement to Reduce Routing Congestion", *U.S. Patent 7,921,392*, Apr. 2011.

[42] P. Saxena, V. Khandelwal, C. Qiao, P-H. Ho, J. C. Lin and M. A. Iyer, "Interconnect-driven Physical Synthesis using Persistent Virtual Routing", *U.S. Patent 7,853,915*, December 2010.

[43] http://www.geosteiner.com/

[44] L. He, C.-K. Koh, D. Z. Pan and X. Yuan, *TRIO release B 1.0*, http://vlsicad.eecs.umich.edu/BK/Slots/cache/cadlab.cs.ucla.edu/software_release/trio/htdocs

[45] C.-W. A. Tsao, *BST-DME source code*, MARCO GSRC Bookshelf, 2002. https://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/BST/download/

[46] ITRS 2013 Edition Report - Interconnect. https://www.semiconductors.org/clientuploads/Research_Technology/ITRS/2013/2013Interconnect.pdf

[47] IBM ILOG CPLEX. www.ilog.com/products/cplex/

APPENDIX: OPTIMAL COST-SKEW TRADEOFF IN ONE DIMENSION

It is intuitively clear that the optimal cost-skew tradeoff curve is monotone: with decreasing skew bound B, the minimum bounded-skew tree cost is non-decreasing. However, the nature of skew allocation within optimal BSTs (i.e., how skew is budgeted across levels in the topology of optimal trees), or how pointset attributes affect the shape of the tradeoff curve, is unknown. Further, the ILP presented above has limited scalability.

To obtain additional insights into the structure of optimal BST solutions for larger instances, we have studied the *one-dimensional* (1-D) bounded-skew tree problem, where all terminals have integer coordinates and are on the x-axis.[2] We implement a dynamic programming algorithm that finds optimal BST solutions for a given 1-D input and skew bound. Results support the "folklore" intuition that skew is best budgeted toward the lower levels of an optimal BST; we also obtain examples of optimal cost-skew tradeoff curves for qualitatively different types of pointsets.

1D-BST Algorithm

A dynamic programming algorithm to find the optimal 1D-BST for a pointset is formulated as follows.

- The input is a pointset $P = (p_1, \ldots, p_n)$ with each $p_i = (x_i, 0)$. p_1 is the source and the remaining p_i are sinks. We must reach all sinks from the source, such that the difference between maximum and minimum source-to-sink distances in the tree is less than or equal to a skew bound B, and total edge cost is minimized.
- Define $x_{min} = \min(x_2, \ldots, x_n)$ and $x_{max} = \max(x_2, \ldots, x_n)$. For the algorithm to maintain an optimal cost-skew tradeoff curve, it must be the case that the source location x_1 lies between x_{min} and x_{max}. Without loss of generality, we let $x_{min} = 1$ in order to simplify the recurrence.
- Let $span(P) = x_{max} - x_{min}$. Let $M[l][s][sh][B]$ be the total cost of the minimum-cost tree over points in P such that $x_i \in [sh{+}1, l{+}sh]$ with source s and skew bound B. $M[l][s][sh][B]$ is computed for each $l \in \{x_{min}, x_{min}+1, \ldots, x_{max}\}$, $sh \in \{1, 2, \ldots, x_{max} - l\}$, $s \in \{sh, sh+1, \ldots, sh+l\}$ and $B \in \{0, 1, \ldots, l\}$.
- Let $S[l][s][sh][B]$ be the (ch, c) pair of the optimal tree of cost c containing sinks p_i such that $sh + 1 \le x_i \le sh + l$

[2]For the 1-D BST problem to be meaningful, edges of a routing tree over the terminals should not be superposed: internal nodes of the tree and embeddings of tree edges are assumed to be displaced very slightly off the x-axis as necessary to avoid overlaps.

and skew bound B where ch is the only child of the source s. (Here, sh refers to a "shift', as explained in the next paragraph.) $S[l][s][sh][B]$ is computed for each $l \in \{x_{min}, x_{min}+1, \ldots, x_{max}\}$, $s \in \{-x_{max}, -x_{max}+1, \ldots, x_{max}\}$, $sh \in \{1, 2, \ldots, x_{max} - l\}$ and $B \in \{0, 1, \ldots, l\}$.

The base cases used to compute M and S are given by Equations (11) and (12), respectively. In these Equations, define $I(p) = 1$ if p is true and $I(p) = \infty$ if p is false. In the case of the M matrix, each sink is connected to an internal node directly above it and in the case of the S matrix,

The recurrences used to compute M and S are given by Equations (13) and (14), respectively. The intuition behind Equation (13) is that any tree of span l is composed of two left and right subtrees with span L and $R = l - L$ respectively. $S[L][s-sh][sh][B].c$ is the cost of the left tree and $S[l-L][s-sh-L][sh+L][B].c$ is the cost of the right tree with source at location s. In each subproblem, we adopt the notational convention that all sinks are shifted by distance sh to the left. So, to ensure that the source location (in a subproblem) is correct with respect to the original problem instance, $M[l][s][sh][B]$, the left subtree's source location will be at $s - sh$ and the right subtree's source location will be at $s - sh - L$.

$$M[1][1][sh][0] = I(x_{sh} \in \{p_2, p_3, \ldots, p_n\}) \tag{11}$$

$$S[1][s][sh][0] = I(x_{sh} \in \{p_2, p_3, \ldots, p_n\}(s-1) \tag{12}$$

$$M[l][s][sh][B] = \tag{13}$$

$$\min_{L} S[L][s-sh][sh][B].c + S[l-L][s-sh-L][sh+L][B].c$$

$$S[l][s][sh][B] = \min_{ch \in \{1,\ldots,l\}} (M[l][ch][sh][B] + |s - ch + sh|) \tag{14}$$

Illustrations. Figure 5 shows how the M and S matrices are filled. The subtree rooted at I_1 contains sinks p_2 and p_3. For $M[l][s][sh][B]$ the values of $(l, s, sh, B) = (3, 2, 0, 0)$. This is due to sinks p_2 and p_3 having their x-coordinates in the window $[sh+1, l+sh] = [1, 3]$. Note that this same subtree has the cost in $M[7][2][0][0]$ as no other sinks aside from p_2 and p_3 are within $[1, 7]$. The subtree rooted at I_2 is held in the solution given by $M[2][1][7][0]$. Both sinks have x-coordinates in $[8, 9]$ and the source of this subtree. For these sinks, we use a "shift" of 7, meaning that each sink is shifted by 7 units to the left. Because of this, it is as though p_4 is at x-coordinate 1 and p_5 is at x-coordinate 2. Due to this shifting, the source, I_2, is now at x-coordinate 1. Lastly, notice that one unit of wire is "wasted" on the edge from I_2 to p_5 in order to preserve a zero skew tree. Wire will be "wasted" anytime two subproblems have maximum and minimum source to sink distances that do not preserve the given skew bound when they are combined into a single tree.

In order to find the final cost of the optimal tree, $M[9][5][0][0]$, we will sweep $L \in \{1, 2, \ldots, 8\}$. Suppose $L = 7$, by Equation 13, $M[9][5][0][0] = S[7][5][0][0] + S[2][-2][7][0]$. Notice how these have already been computed in previous steps so we can used our cached solutions in order to find the minimum cost bounded skew tree.[3]

In addition to these recurrences, the following cases for M will need to be considered: (i) the trees with one child of the source and (ii) the minimum spanning trees with source locations in $[sh+1, l+sh]$. We need to consider case (i) because our recurrence is defined when an internal node has two children – but this may not necessarily be the case. Case (ii) is necessary because without it, sinks would not be able to add edges to other sinks. The base case for the M tensor is simply that each sink can be connected to a source at the same location for 0 cost and 0 skew.

[3]As mentioned previously, wire may need to be "wasted" on one of the subtrees in order to preserve the skew bound.

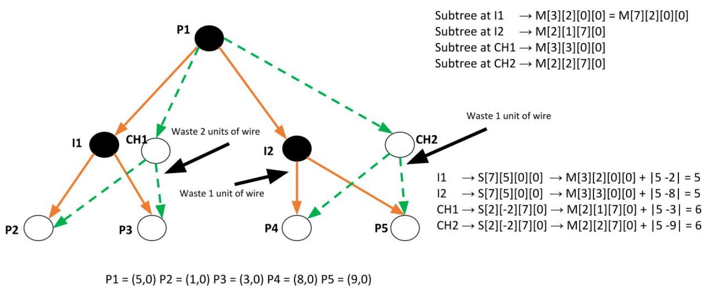

P1
I1
CH1
I2
CH2
P2
P3
P4
P5
Waste 2 units of wire
Waste 1 unit of wire
Waste 1 unit of wire
Subtree at I1 → M[3][2][0][0] = M[7][2][0][0]
Subtree at I2 → M[2][1][7][0]
Subtree at CH1 → M[3][3][0][0]
Subtree at CH2 → M[2][2][7][0]
I1 → S[7][5][0][0] → M[3][2][0][0] + |5 -2| = 5
I2 → S[7][5][0][0] → M[3][3][0][0] + |5 -8| = 5
CH1 → S[2][-2][7][0] → M[2][1][7][0] + |5 -3| = 6
CH2 → S[2][-2][7][0] → M[2][2][7][0] + |5 -9| = 6
P1 = (5,0) P2 = (1,0) P3 = (3,0) P4 = (8,0) P5 = (9,0)

Figure 5: Example of how M and S are constructed over a pointset P.

Empirical Studies

We now briefly describe example insights that can be obtained in a quantified manner using the DP for 1-D BST. A first insight concerns the impact of sink clustering. To create diverse pointsets as inputs to the DP algorithm, we define a *cluster* in a pointset to be any set of sinks with span equal to the number of sinks in the cluster. In other words, a cluster is a set of 'contiguous' points. We construct pointsets that have c clusters evenly spaced within the span of the pointset.

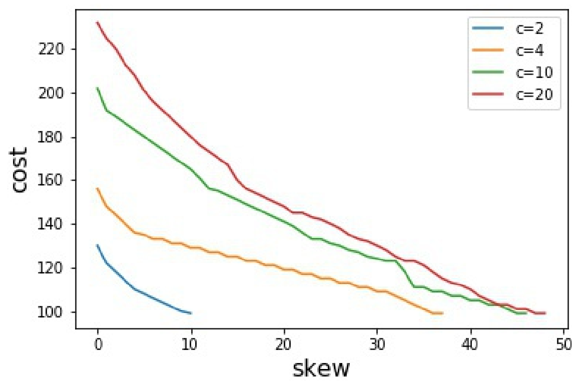
As the number of clusters in a pointset increases, internal nodes of the bounded-skew tree will need to be spread out more at lower levels of the tree in order to reduce skew. For example, with only two clusters, we might satisfy a given skew bound by constructing two subtrees for each cluster, but having 16 clusters would require creation of 16 trees and addition of more internal nodes to connect back to the source.

Figure 6 shows that for a given fixed value of B, the optimum tree cost will increase as the number of clusters increases. This essentially visualizes the benefits seen in use of multi-bit flip-flops and other clock sink clustering techniques, as a function of skew bound. We also observe that the cost impact of clustering depends on the overall density of sinks in the placement. The spread between the cost-skew tradeoff curves is much greater in the top figure ($n = 20$) than in the bottom figure ($n = 80$).

The 1-D DP solutions also improve our intuition regarding optimal BST cost as a function of the number of sinks and of the skew bound B. Figure 7 shows how – for a fixed span – the tree cost increases with the number of sinks.[4] Additionally, the cost-skew tradeoff curve is steeper for small values of skew: when the skew bound B is small, an increase in skew will decrease cost more rapidly than if the skew were already large. Thus, we can see the wirelength and dynamic power returns from relaxation of skew constraints.

A third example insight from 1-D BST concerns the classic question of how available skew should be optimally budgeted across levels of a clock tree. Figure 8 shows internal nodes and the cumulative source-to-sink distance at each internal node. In the figure,

[4]While, e.g., a square-root relationship between ZST cost and n has been conjectured in the past (cf. the theory of subadditive functionals of planar pointsets), no formal result has been established.



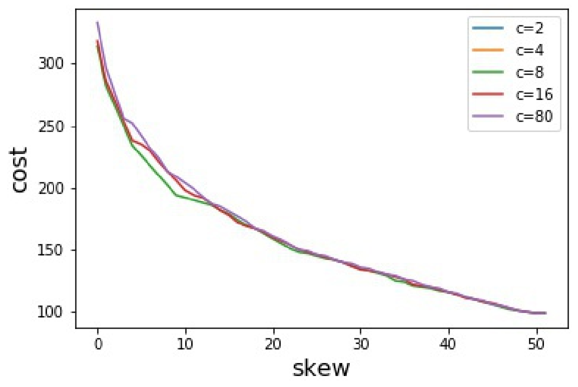
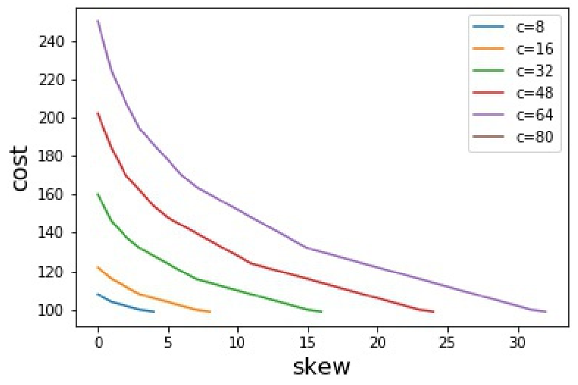


Figure 6: Effect of the number of clusters, c, on cost-skew tradeoff curve for $(span(P), n) = (100, 20)$ (top) and $(span(P), n) = (100, 80)$ (bottom).

sinks are displayed in black at the bottom, and internal nodes are colored according to their source-to-sink distance. The pointset consists of 80 sinks with a span of 100 and 40 clusters with the source at location $x = 32$. DP solutions show that skew is preserved in favor of cost at high levels of the tree (near the root), and cost is preserved in favor of skew at lower levels of the tree (near the leaves). By the time the internal node that is lowest in the tree is reached, nodes along the same level of the tree have the same source-to-sink distance. This is most apparent when the tree elects to waste wirelength in order to maintain a balance among the internal nodes in higher levels of the tree. For example, in Figure 8, the path created from nodes (a-b-c-d) wastes wirelength as the path could have been taken from (a-d). This is accomplished in order to preserve a similar source-to-sink distance among internal nodes in the tree at this level. This will ensure that sinks that are far away from the source share a similar source to sink distance. Once the lowest level of the tree is reached, we see that the internal nodes tend to be sparse relative to the sinks surrounding them. This means that the sinks are reached from other sinks, i.e., these subtrees rooted at internal nodes at the bottom of the tree have high skew. This implies that skew is budgeted to be spent at lower levels of the tree rather than higher levels of the tree.



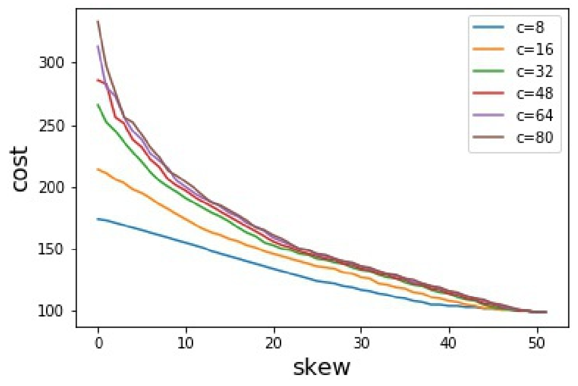


Figure 7: Effect of n on cost vs. skew tradeoff curve for $(span(P), c) = (100, 2)$ (top) and $(span(P), c) = (100, n)$ (bottom).

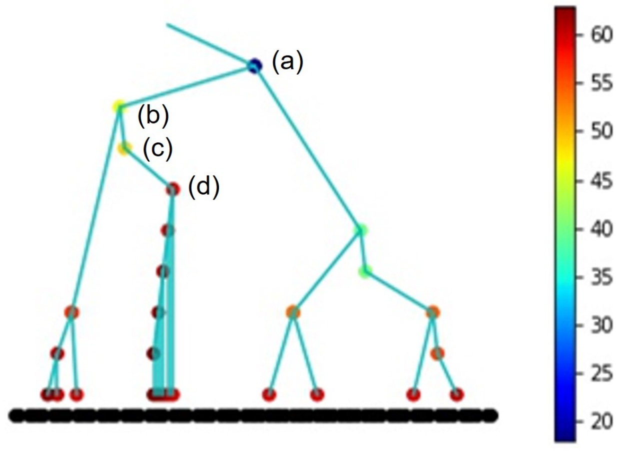


Figure 8: Optimal BST topology with $span(P) = 100$, $s = 32$ and $skew = 8$. Each internal node in the tree is colored with its source-to-sink distance. Sinks are colored black.