SALT: Provably Good Routing Topology by a Novel Steiner Shallow-Light Tree Algorithm

Gengjie Chen, Peishan Tu, Evangeline F. Y. Young

Department of Computer Science & Engineering
The Chinese University of Hong Kong

Nov 15, 2017

香港中文大學
The Chinese University of Hong Kong

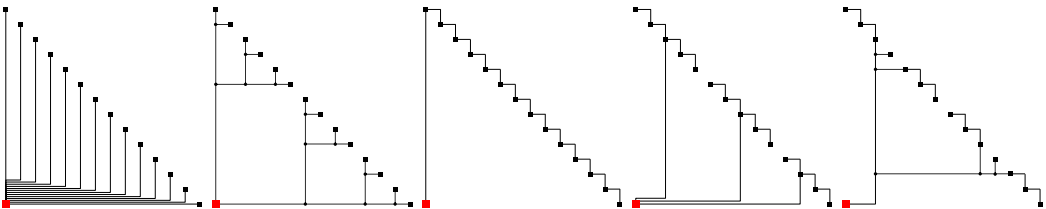
Introduction

- Timing and power are crucial in chip design.
- In routing tree:
  - **Path length** implies wire delay;
  - **Tree weight** implies routing resource usage (routability), power consumption, cell delay and wire delay.
- In spanning/Steiner $(\bar{\alpha}, \bar{\beta})$-shallow-light tree (SLT) T:
  - **Shallowness** $\alpha = max\{\frac{d_T(r,v)}{d_G(r,v)} | v \in V \backslash \{r\}\} \leq \bar{\alpha}$.
    - $d_G(r, v)$: distance from v to root r on graph/metric G.
  - **Lightness** $\beta = \frac{w(T)}{w(MST(G))} \leq \bar{\beta}$.

Introduction

	shallowest	lightest	shallow light
spanning	spanning SPT* ($O(m + n \log n)$)	MST ($O(m + n \log n)$)	spanning SLT
Steiner	Steiner SPT (NP hard)	SMT (NP hard)	**Steiner SLT**
rectilinear Steiner	RSMA† (NP hard)	RSMT (NP hard)	**rectilinear Steiner SLT**

*shortest-path tree †rectilinear Steiner minimum arborescence



(a) Spanning SPT ($\alpha = \frac{13}{13}, \beta = \frac{182}{39}$) (b) RSMA ($\alpha = \frac{13}{13}, \beta = \frac{54}{39}$) (c) RMST/RSMT ($\alpha = \frac{39}{13}, \beta = \frac{39}{39}$) (d) Spanning SLT ($\alpha = \frac{17}{13}, \beta = \frac{61}{39}$) (e) Steiner SLT ($\alpha = \frac{17}{13}, \beta = \frac{44}{39}$)

Introduction

Previous Work

- Spanning $(1+\epsilon, O(\frac{1}{\epsilon}))$-SLT
  - ABP/BRBC $(1+2\epsilon, 1+\frac{2}{\epsilon})$ [Awerbuch, TR'91] [Cong, TCAD'92];
  - KRY $(1+\epsilon, 1+\frac{2}{\epsilon})$ [Khuller, SODA'93, Algorithmica'95].
- Steiner $(1+\epsilon, O(\log \frac{1}{\epsilon}))$-SLT
  - ES $(1+2\epsilon, 4+2\lceil \log \frac{2}{\epsilon} \rceil)$ [Elkin, FOCS'11, SICOMP'15].
- PD combines SPT and MST [Alpert, TCAD'95].
- Bonn trades off between cell and wire delay [Scheifele, ICCAD'16, Algorithmica'17].

Introduction

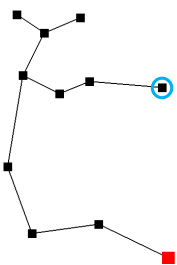
Major Contributions

- Propose **SALT** for general-graph Steiner SLT, whose shallowness-lightness bound is $(1+\epsilon, 2+\lceil \log \frac{2}{\epsilon} \rceil)$.
- Reduce **runtime** from $O(n^2)$ to $O(n \log n)$ in Manhattan space.
- Integrate SALT with classical RSMA and RSMT algorithms, which provides a smooth **trade-off** between RSMA and RSMT.
- Propose several effective **post processing** methods.

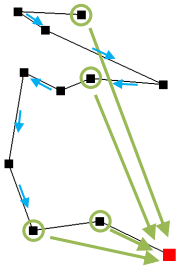
Outline

Steiner SLT Algorithm (SALT)

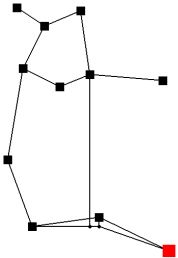
Preliminary: ES algorithm



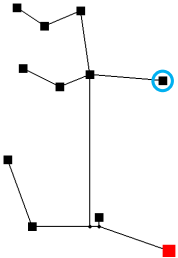
(a) MST T_M



(b) Path P



(c) Graph $T_M \cup T_B$

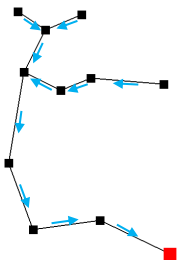


(d) ES T

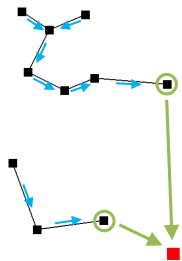
- Construct MST T_M.
- Identify **breakpoints** B on **Hamiltonian path** P.
- Obtain **Steiner SPT** T_B on $G[B \cup \{r\}]$, and get graph $T_M \cup T_B$.
- Construct spanning SPT on $T_M \cup T_B$, which is the output T.

Steiner SLT Algorithm (SALT)

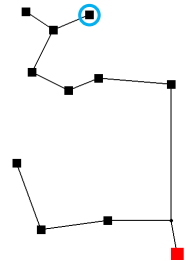
Framework



(a) MST T_M



(b) Forest F

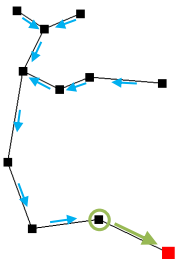


(c) SALT T

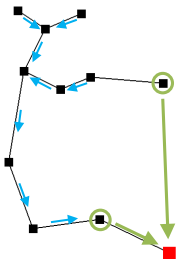
- Construct **MST** T_M.
- Identify **breakpoints** B during **DFS** on T_M, which results to forest F.
- Obtain **Steiner SPT** T_B on $G[B \cup \{r\}]$, and $T = F \cup T_B$ is the output.

Steiner SLT Algorithm (SALT)

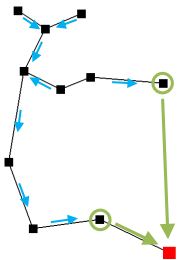
DFS & Breakpoints



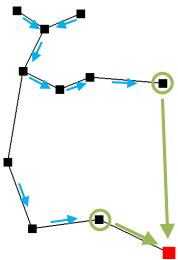
(a) Initial



(b) Path length improved



(c) Further improved



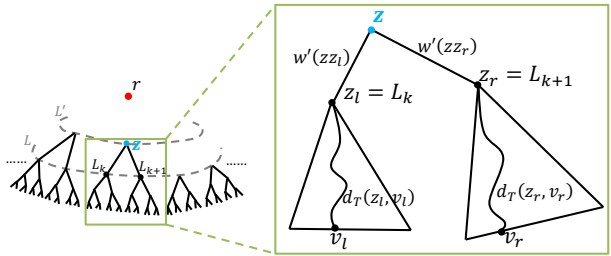
(d) Final

Make sure $d_T(r, v) \leq \bar{\alpha} \cdot d_G(r, v)$.

- Breakpoints will be connected to r by shortest paths.
- Other vertexes also benefit.

Steiner SLT Algorithm (SALT)

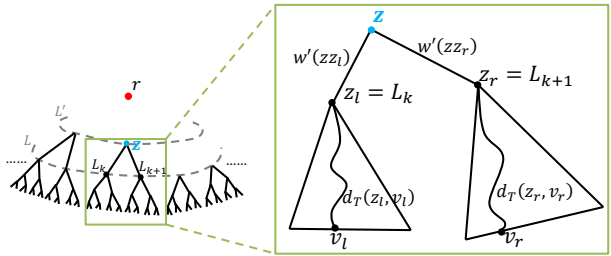
Light Steiner SPT

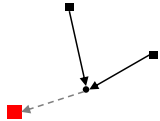


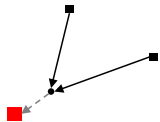
- A full balanced binary tree.
- Constructed level by level from bottom.
- Merge neighboring vertexes pair by pair into Steiners in each level.
  - Determine Steiner by minimizing edge weights while preserving shortest paths.
  - Select a light matching for paring up along (Hamiltonian) circle.

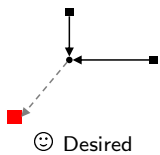
Steiner SLT Algorithm (SALT)

Light Steiner SPT





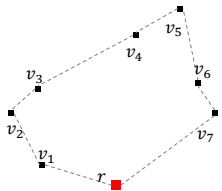




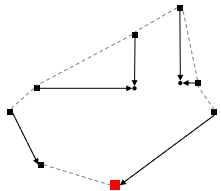
- A full balanced binary tree.
- Constructed level by level from bottom.
- Merge neighboring vertexes pair by pair into Steiners in each level.
  - Determine Steiner by minimizing edge weights while preserving shortest paths.
  - Select a light matching for paring up along (Hamiltonian) circle.

Steiner SLT Algorithm (SALT)

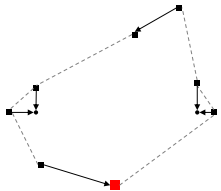
Light Steiner SPT (Cont.): a Manhattan Example



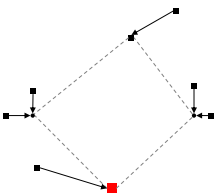
(a) Level 1



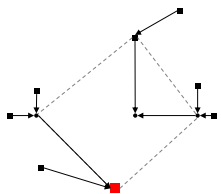
(b) Bad matching



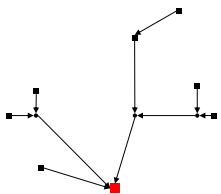
(c) Level 2



(d) Level 2



(e) Level 3



(f) Level 4

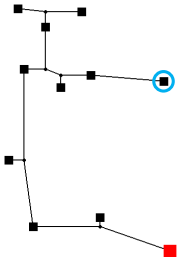
Steiner SLT Algorithm (SALT)

Key Facts

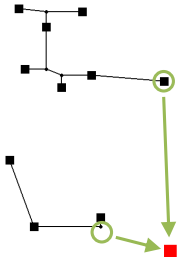
- Three differences compared to ES:
  - Tighter criterion for breakpoints;
  - Better initial topology (MST instead of Hamiltonian path);
  - Much lighter Steiner SPT (with lightness bound $\bar{\beta} = \lceil \log n \rceil$).
    - ES: $1 + 2\lceil \log n \rceil$.
- SALT generates a Steiner $(1 + \epsilon, 2 + \lceil \log \frac{2}{\epsilon} \rceil)$-SLT.
  - ES: $(1 + 2\epsilon, 4 + 2\lceil \log \frac{2}{\epsilon} \rceil)$.

Rectilinear Steiner SLT Algorithm (Rectilinear SALT)

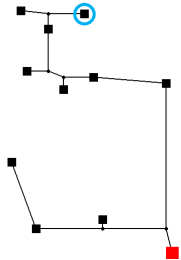
Framework



(a) RSMT T_M by FLUTE



(b) Forest F



(c) Rectilinear SALT T

- Construct **RSMT** T_M by FLUTE [Chu, TCAD'08].
- Get breakpoints B and forest F.
- Obtain **RSMA** T_B on $G[B \cup \{r\}]$ by CL [Cordova, TR'94], and $T = F \cup T_B$ is the output.

Rectilinear Steiner SLT Algorithm (Rectilinear SALT)

Key Facts

- Two differences compared to SALT:
  - Better initial topology (RSMT by FLUTE instead of MST);
  - Lighter Steiner SPT (RSMA by CL).
- Improve shallowness α and lightness β in practice.
- Very efficient: $O(n \log n)$ time.

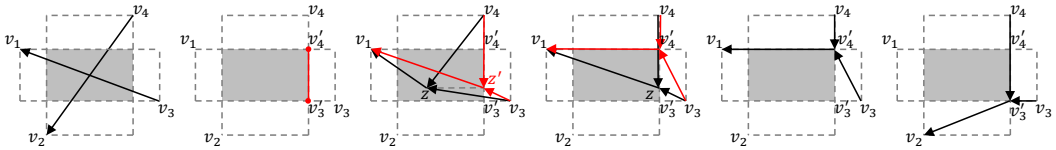
Post Processing

Three post processing techniques

- Canceling intersected edges
- L-shape flipping
- U-shape shifting

Post Processing

Canceling Intersected Edges

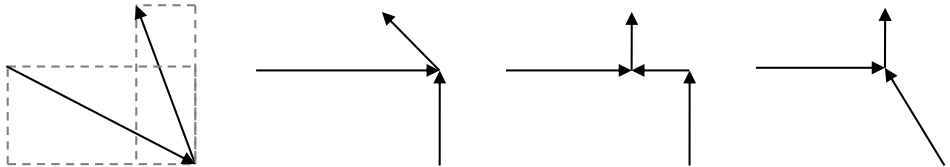


(a) Intersection box (b) Child corners v'_3, v'_4 (c) z should be on edge $v'_3v'_4$ (d) z should be either v'_3 or v'_4 (e) 1st Solution (f) 2nd Solution

- Improve (i) path length, (ii) wirelength.
- Efficiently identified by R-tree.
- Best Steiner vertex z should be a **child corner** of intersection box.
  - Child corner: the corner closest to a child vertex among four.

Post Processing

L-Shape Flipping



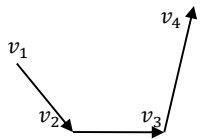
(a) Input (b) First L-shape flipping (c) Second L-shape flipping (d) Removing redundancy

Z-shape flipping by iterative L-shape flipping.

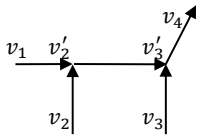
- Improve (i) path length, (ii) wirelength.
- Optimal by dynamic programming [Ho, TCAD'90].
- $O(n)$ due to bounded vertex degree in SALT.
- Iterate until no improvement.

Post Processing

U-Shape Shifting



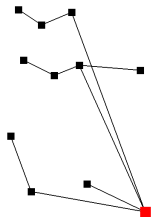
(a) Input



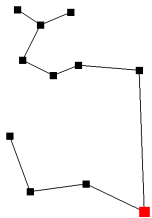
(b) Output

- Improve (i) path length, (ii) wirelength, (iii) Elmore delay [Boese, DAC'93].

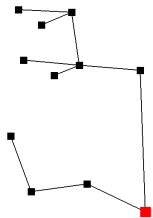
Experimental Results



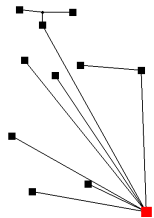
(a) ABP



(b) KRY



(c) PD



(d) Bonn

Sample runs of various algorithms ($\epsilon = 1$)

- i.e., $\bar{\alpha} = 1 + 2\epsilon = 3$ for ABP/BRBC, $\bar{\alpha} = 1 + \epsilon = 2$ for KRY & PD.
- ABP/BRBC ($\alpha = 1.90$, $\beta = 1.35$);
- KRY ($\alpha = 1.43, \beta = 1.10$);
- PD ($\alpha = 1.11, \beta = 1.15$);
- Bonn ($\alpha = 1.22, \beta = 2.25$).

Experimental Results

Table: ICCAD 2015 Benchmark Statistics

Design	#cells ($\times 10^3$)	#nets classified by pin number ($\times 10^3$)						
		2	3–9	10–19	20–29	30–39	≥ 40	≥ 3
superblue1	1932	893	281	23	11	6	0.9	323
superblue3	1876	952	215	35	15	6	1.1	273
superblue4	796	610	162	17	9	4	0.5	192
superblue5	982	824	242	18	8	5	0.7	273
superblue7	768	1493	338	63	27	11	1.7	441
superblue10	1087	1457	385	31	14	9	1.2	441
superblue16	1213	756	213	17	7	5	0.3	243
superblue18	1210	575	156	24	11	5	0.6	197
Total	9863	7559	1992	229	103	51	7.0	**2382**

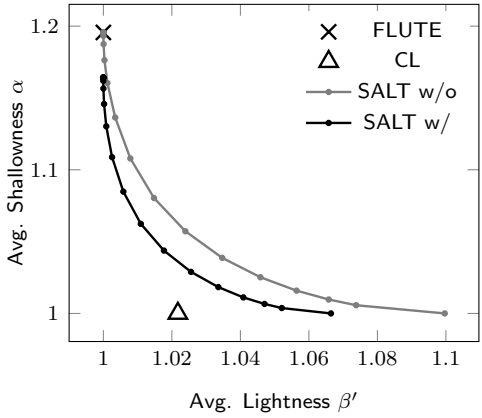
- ICCAD 2015 Contest benchmarks with 2.4 million nets (excluding 2-pin nets).

Experimental Results

- ϵ is set to 20 values ranging from 0 to 73.895.
- Three metrics for each tree:
  - Shallowness α;
  - Lightness $\beta' = \frac{w(T)}{w(FLUTE)}$ (instead of $\beta = \frac{w(T)}{w(MST)}$);
  - Delay γ = longest Elmore delay among all paths, normalized by a lower bound.

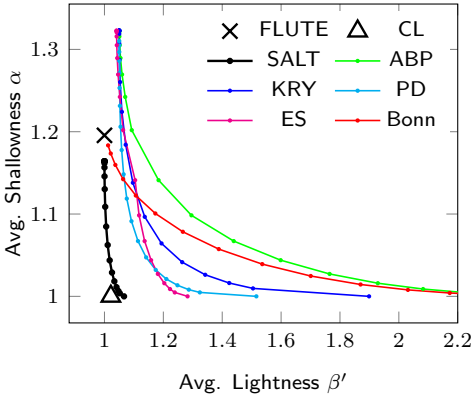
Experimental Results

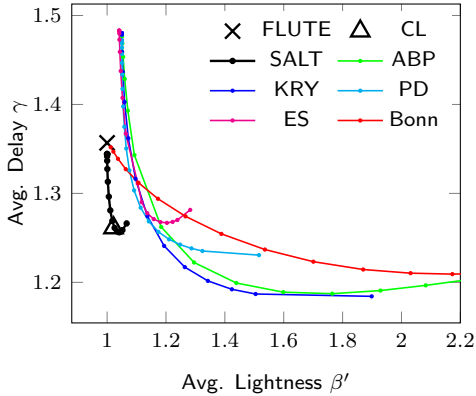
	SALT w/o post proc.			SALT w/ post proc.		
ϵ	β'	α	γ	β'	α	γ
0.000	1.100	1.000	1.271	**1.066**	**1.000**	**1.266**
0.050	1.074	1.006	1.258	**1.052**	**1.004**	**1.259**
0.075	1.066	1.010	1.256	**1.047**	**1.007**	**1.257**
0.113	1.056	1.016	1.256	**1.041**	**1.011**	**1.256**
0.169	1.046	1.025	1.258	**1.034**	**1.018**	**1.257**
0.253	1.035	1.039	1.263	**1.026**	**1.029**	**1.261**
0.380	1.024	1.057	1.273	**1.018**	**1.044**	**1.269**
0.570	1.015	1.080	1.287	**1.011**	**1.062**	**1.281**
0.854	1.008	1.108	1.305	**1.006**	**1.085**	**1.296**
1.281	1.003	1.136	1.323	1.003	**1.109**	**1.313**
1.922	1.001	1.160	1.339	1.001	**1.130**	**1.328**
2.883	1.000	1.176	1.349	1.000	**1.146**	**1.337**
4.325	1.000	1.187	1.354	1.000	**1.157**	**1.342**
6.487	1.000	1.193	1.356	1.000	**1.162**	**1.344**
9.731	1.000	1.195	1.357	1.000	**1.164**	**1.344**
...	1.000	1.196	1.357	1.000	**1.164**	**1.344**



- Post proc. simultaneously improves shallowness α, lightness β' and delay γ.
- Efficient: routing + post proc. on 2.4 million nets for 20 times in 22.5 min.

Experimental Results





- Dominate other methods in shallowness-lightness trade-off.
- Good in delay-lightness trade-off.
- No parallel edges.

Conclusions

Conclusions

- Steiner $(1+\epsilon, 2+\lceil \log \frac{2}{\epsilon} \rceil)$-SLT for general-graph.
- Reduce $O(n \log n)$ runtime in Manhattan space.
- Integration with classical RSMA and RSMT algorithms.
- Effective post processing methods.

Further work

- Be closer to RSMA for small ϵ.
- Consider routing congestion / blockage.