# Separator-based Pruned Dynamic Programming for Steiner Tree

Yoichi Iwata (NII)

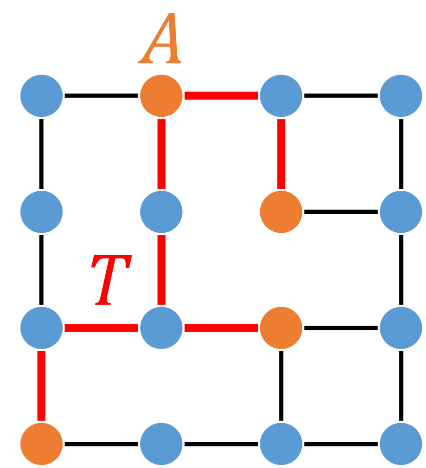
Takuto Shigemura (U-Tokyo)

appeared in AAAI 2019

https://www.aaai.org/Papers/AAAI/2019/AAAI-IwataY.4578.pdf

# Steiner Tree Problem

**Steiner tree problem**
Input: graph $G$, terminals $A \subseteq V(G)$
Output: minimum-weight tree connecting $A$




Various applications:

- ✓ VLSI design
- ✓ fiber-optic network design
- ✓ team formulation in social networks

# PACE Challenge 2018

https://pacechallenge.wordpress.com/pace-2018/

**Track 1:** small number of terminals ($|A| \leq 136$)

✓ $O\left(3^{|A|}n + 2^{|A|}(m + n\log n)\right)$ by DP [Erickson-Monma-Veinott 87]

✓ EMV + $A^*$-search [Hougardy-Silvanus-Vygen 17]

**Track 2:** small tree-width ($w \leq 47$)

✓ $w^{O(w)}m$ by DP

✓ $2^{O(w)}m$ by a rank-based DP [Bodlaender-Cygan-Kratsch-Nederlof 15]

**Track 3:** heuristic

# Results

https://pacechallenge.org/files/PACE18-report.pdf

**Track 1:** small number of terminals

1. Our team solved 95/100 by *pruned DP*
2. Maziarz and Polak solved 94/100 by HSV
3. Koch and Rehfeldt solved 93/100 by ILP

**Track 2:** small tree-width

1. Koch and Rehfeldt solved 92/100 by ILP
2. Our team solved 77/100 by *pruned DP* + $w^w$-DP
3. Tom van der Zanden solved 58/100 by rank-based DP

for public instances:

pruned DP alone solved 81/100
$w^w$-DP alone solved 44/100
combined solved 84/100

# Outline

1. EMV Algorithm
2. Separator-based Pruning
3. Other techniques
4. Experiments

# Dynamic Programming [Dreyfus-Wagner 71]

For $S \subseteq V$, let $\mathrm{opt}(S) :=$ weight of min Steiner tree for terminals $S$.

$$\mathrm{opt}(S \cup \{u\}) = \min \begin{cases} \mathrm{opt}(S' \cup \{u\}) + \mathrm{opt}\big((S \setminus S') \cup \{u\}\big) & (S' \subseteq S) \\ \mathrm{opt}(S \cup \{v\}) + w(vu) & \big(vu \in E(G)\big) \end{cases}$$

# EMV Algorithm [Erickson-Monma-Veinott 87]

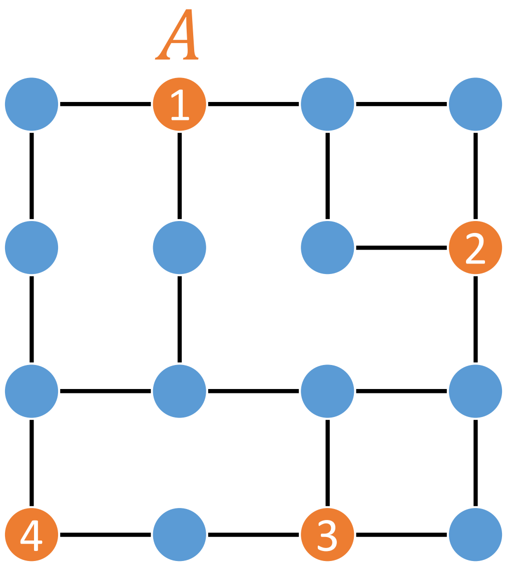
$d(S, u) = \infty$ for $\forall S \subseteq A$ and $\forall u \in V$
$d(\{a\}, a) = 0$ for $\forall a \in A$
for $S \subseteq A$ in ascending order of $|S|$
    update $d(S, u)$ for $\forall u$ by Dijkstra
    for $S' \subseteq A \setminus S$
        update $d(S \cup S', u)$ for $\forall u$

$O\left(3^{|A|} n + 2^{|A|}(m + n \log n)\right)$ time

Can we avoid computing all $d(S, u)$?
$\rightarrow$ Yes, for special instances. [EMV 87]

# Special Case [Erickson-Monma-Veinott 87]

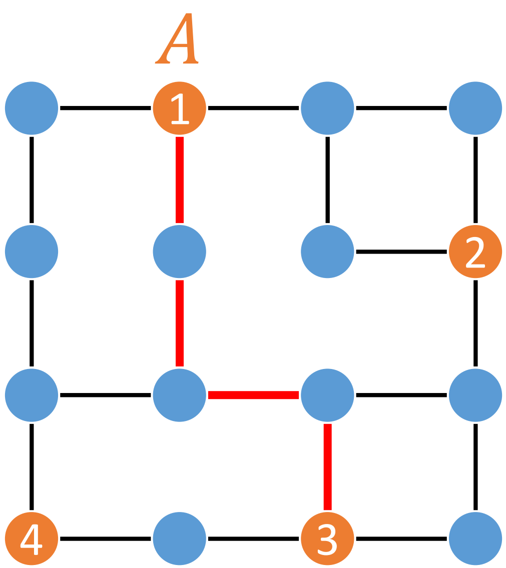
If the graph if planar and all the terminals are on a single face, the running time is improved to $O(|A|^3 n + |A|^2 n \log n)$.



# Special Case [Erickson-Monma-Veinott 87]

If the graph if planar and all the terminals are on a single face, the running time is improved to $O(|A|^3 n + |A|^2 n \log n)$.

But, of course, this is too special to apply in practice…




Steiner tree for $\{1, 3\}$ always *separates* $\{2, 4\}$. So we can skip the computation of $d(\{1,3\},*)$.

In general, we only need to compute

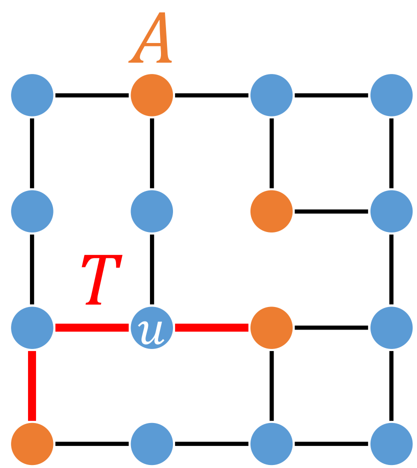
$$d(\{i, i+1, \dots, j\},*)$$

# Outline

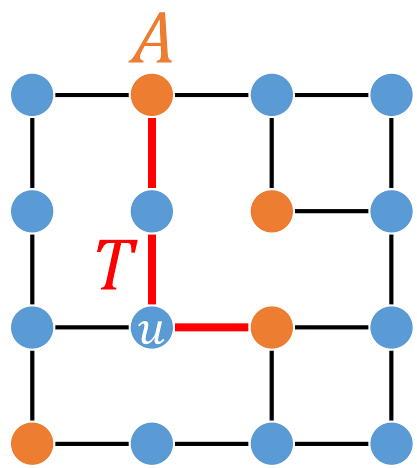
1. EMV Algorithm
2. Separator-based Pruning
3. Other techniques
4. Experiments

# Important Partial Solution

A Steiner tree $T$ for $S \cup \{u\}$ is called *important* if there is a Steiner tree $T'$ for $(A \setminus S) \cup \{u\}$ s.t. $T + T'$ is a minimum Steiner tree for $A$.




important




unimportant

# Important Partial Solution

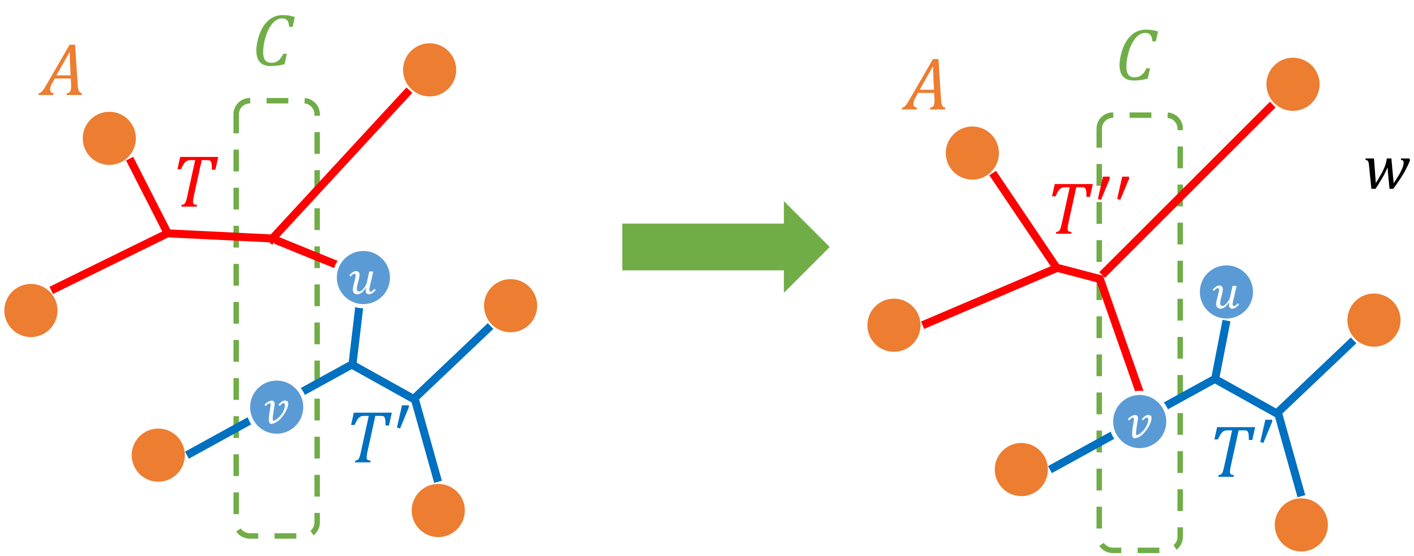
A Steiner tree $T$ for $S \cup \{u\}$ is called *important* if there is a Steiner tree $T'$ for $(A \setminus S) \cup \{u\}$ s.t. $T + T'$ is a minimum Steiner tree for $A$.

In EMV algorithm, we can safely skip computations for unimportant $(S, u)$.

But how can we check the importance?

# Necessary Condition of Importance

**Key Lemma:**
A Steiner tree $T$ for $S \cup \{u\}$ is not important if there is an $(A \setminus S)$-separator $C$ such that for every $v \in C$, there is a Steiner tree for $S \cup \{v\}$ of weight strictly less than the weight of $T$.

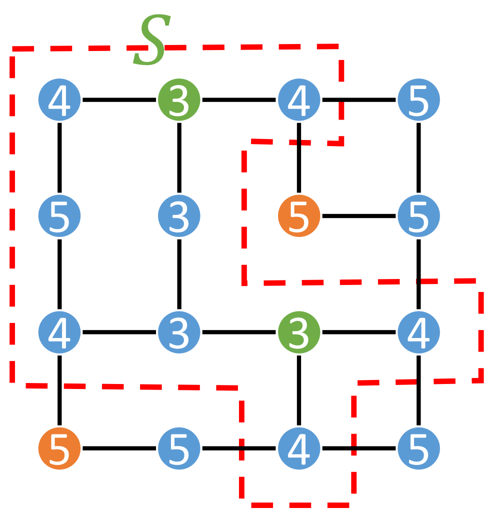



$$w(T'') < w(T)$$

# Separator Construction

After computing $d(S,*)$, we compute the minimum $x$ s.t. $C_x := \{\, u \in V \mid d(S,u) \le x \,\}$ separates $A \setminus S$.

Then, $C_x$ satisfies the condition for every $(S,u)$ with $d(S,u) > x$.



# Pruned DP Algorithm

$P \leftarrow \emptyset$ # set of processed $S$

while $\exists$unprocessed $S$ s.t. $d(S, u) \neq \infty$ for some $u$

pick smallest such $S$

update $d(S, u)$ for $\forall u$ by Dijkstra

compute minimum $x$ s.t. $C_x$ separates $A \setminus S$

drop $(S, u)$ from $d$ for $\forall (S, u)$ s.t. $d(S, u) > x$

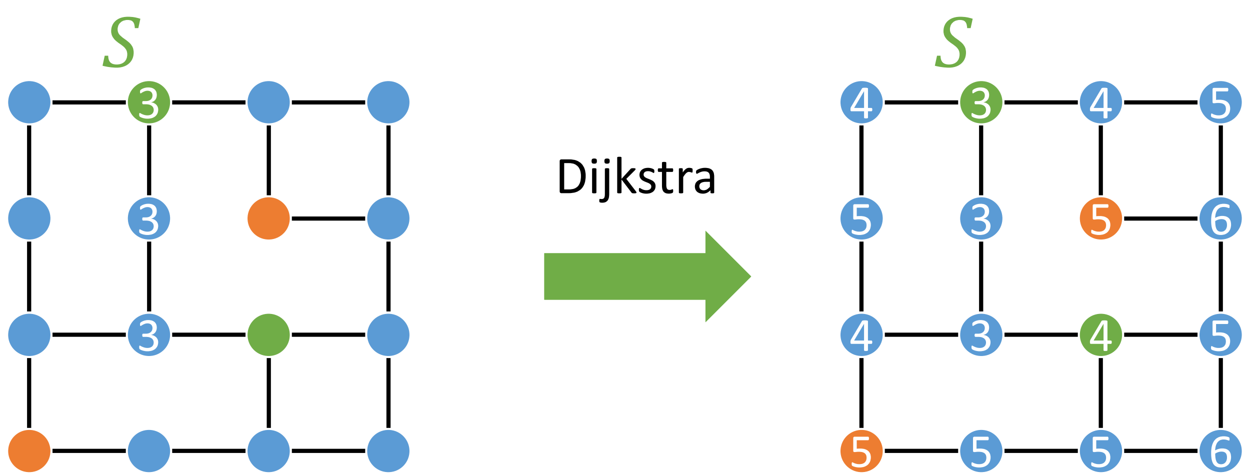
for $S' \in P$ s.t. $S \cap S' \neq \emptyset$

update $d(S \cup S', u)$ for $\forall u$

push $S$ into $P$

# Restore dropped information (1)

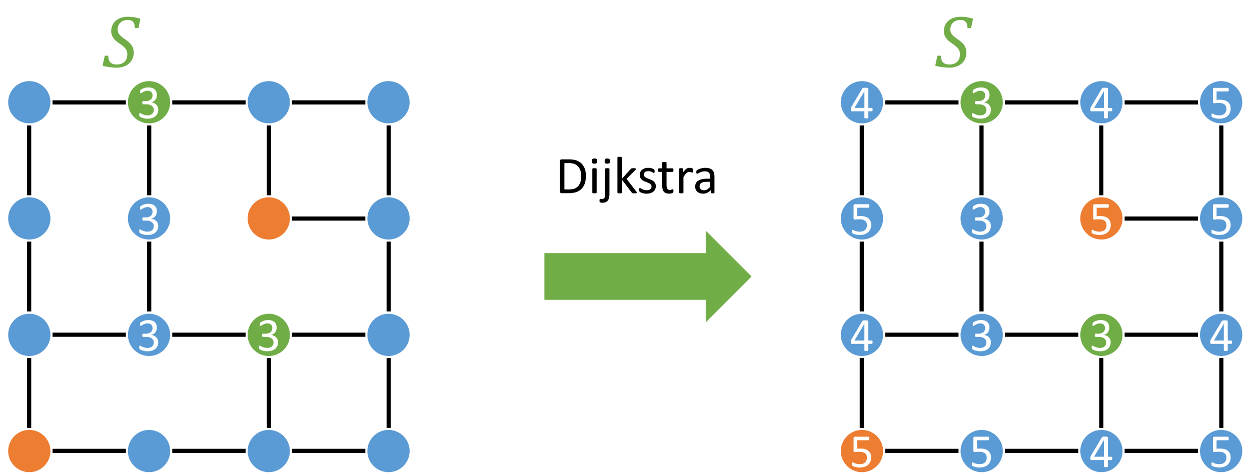
For unimportant $(S, u)$, we may have
$d(S, u) > \mathrm{opt}(S \cup \{u\})$.
This can interfere with the pruning…




$C_4$ does not separate $A \setminus S$ ☹

# Restore dropped information (2)

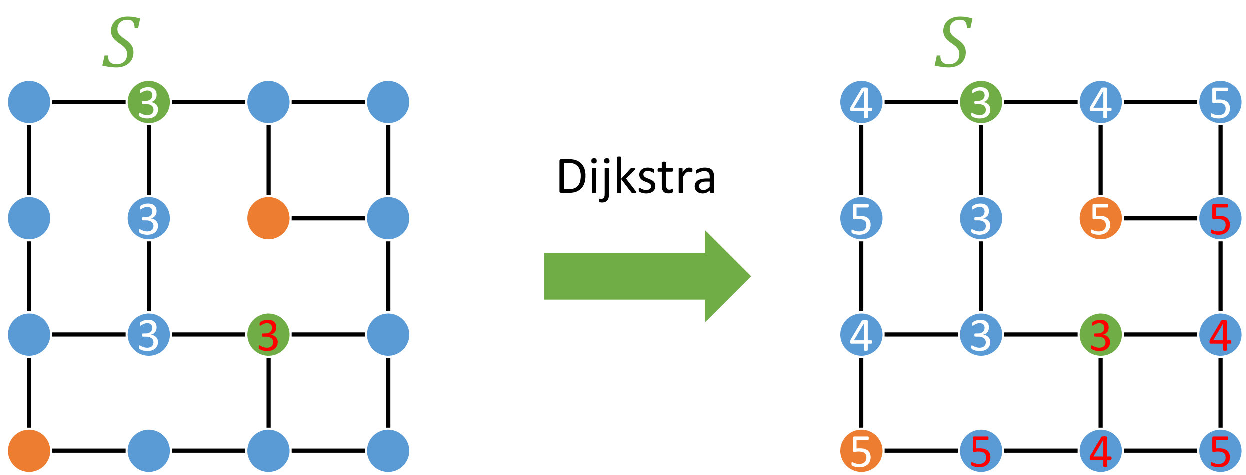
Before running Dijkstra, we partially restore $d(S, u)$ as follows. For each $u$ with $d(S, u) \neq \infty$, we construct the corresponding Steiner tree $T_u$ for $S \cup \{u\}$, and update $d(S, v)$ with $d(S, u)$ for $\forall v \in V(T_u)$.




$C_4$ separates $A \setminus S$ ☺

# Restore dropped information (3)

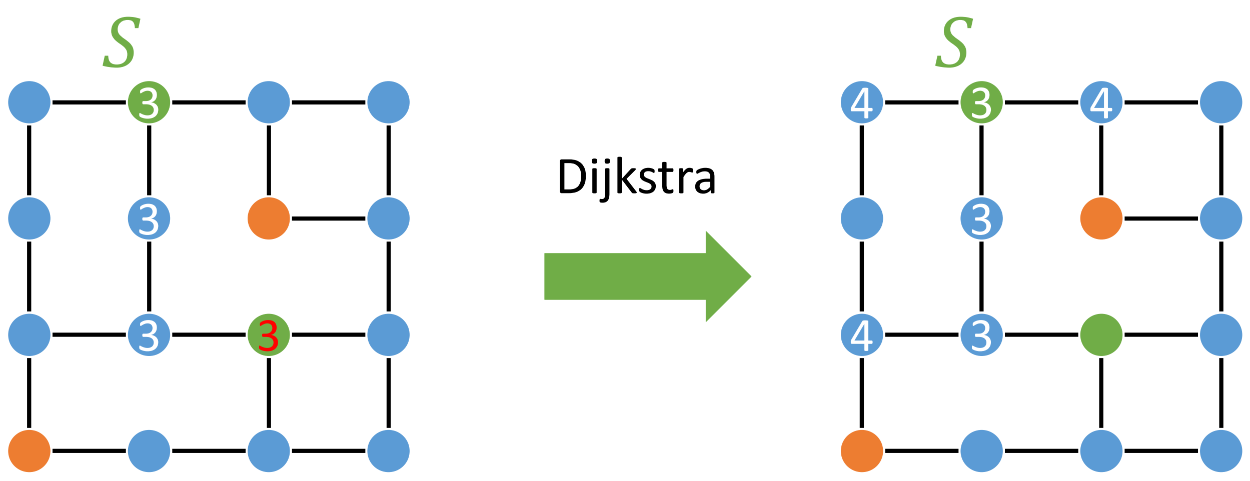
If a Steiner tree $T$ for $S \cup \{u\}$ is unimportant, $T + P_{uv}$ is also unimportant. So we can safely drop such $(S, v)$.




$C_4$ separates $A \setminus S$ ☺

# Restore dropped information (3)

If a Steiner tree $T$ for $S \cup \{u\}$ is unimportant, $T + P_{uv}$ is also unimportant. So we can safely drop such $(S, v)$.




$C_4$ separates $A \setminus S$ ☺

# Outline

1. EMV Algorithm
2. Separator-based Pruning
3. Other techniques
4. Experiments

# Data Structure (1)

$P \leftarrow \emptyset$ # set of processed $S$

while ∃unprocessed $S$ s.t. $d(S, u) \neq \infty$ for some $u$

  …

  for $S' \in P$ s.t. $S \cap S' \neq \emptyset$

    update $d(S \cup S', u)$ for $\forall u$

  push $S$ into $P$

How can we apply the merge operation efficiently?

Let $\mathrm{valid}(S) \coloneqq \{ u \in V \mid d(S, u) \neq \infty \}$.

We want to find all $S'$ such that

1. $S \cap S' = \emptyset$ and
2. $\mathrm{valid}(S) \cap \mathrm{valid}(S') \neq \emptyset$

# Data Structure (2)

We maintain the set of processed $S$ using the following binary trie.

Each leaf $t$ keeps $S_t \subseteq A$.

Each internal node $t$ with leaves $L_t$ keeps

1. $I_i \coloneqq \bigcap_{t \in L_i} S_t$
2. $U_i \coloneqq \bigcup_{t \in L_i} \text{valid}(S_t)$

When searching $S'$, we can stop if $S \cap I_i \neq \emptyset$ or $\text{valid}(S) \cap U_i = \emptyset$.

# Meet in the Middle

Any Steiner tree for $A$ can be written as a sum of three Steiner trees for $S_1, S_2, S_3$ with $1 \le |S_1| \le |S_2| \le |S_3| \le |A|/2$.

1. We can stop after processing all $S$ of size $\le |A|/2$.
2. When merging, we can iterate only over $S'$ of size at most $2|A| - 2|S|$.

$P \leftarrow \emptyset$ # set of processed $S$

while $\exists$unprocessed $S$ s.t. $d(S, u) \neq \infty$ for some $u$

  …

  for $S' \in P$ s.t. $S \cap S' \neq \emptyset$

    update $d(S \cup S', u)$ for $\forall u$

  push $S$ into $P$

# Outline

1. EMV Algorithm
2. Separator-based Pruning
3. Other techniques
4. Experiments

# Environment

**Data set:** from the DIMACS and PACE

**List of Solvers:**

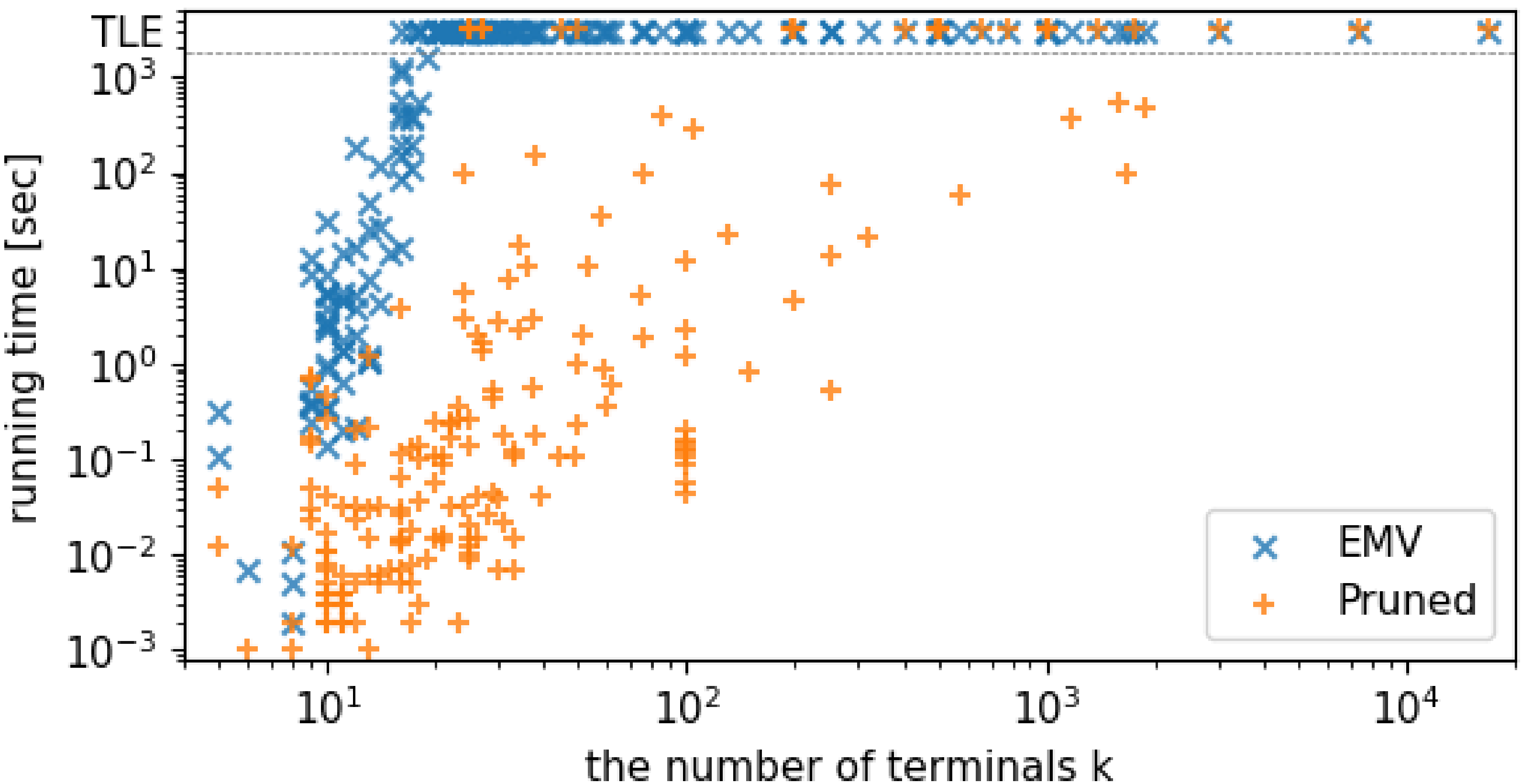
- Pruned: the proposed pruned DP algorithm
- EMV: the classical DP algorithm
- HSV: EMV + $A^*$-search
- SCIP-Jack [Gamrath,Koch,Maher,Rehfeldt,Shinano 17]:
  ILP solver (PACE version)

**Setting:** Intel Xeon E5-2670 (2.6 GHz), single thread, time limit = 30 minutes, memory limit = 6GB

(same as PACE)

# Comparison with EMV

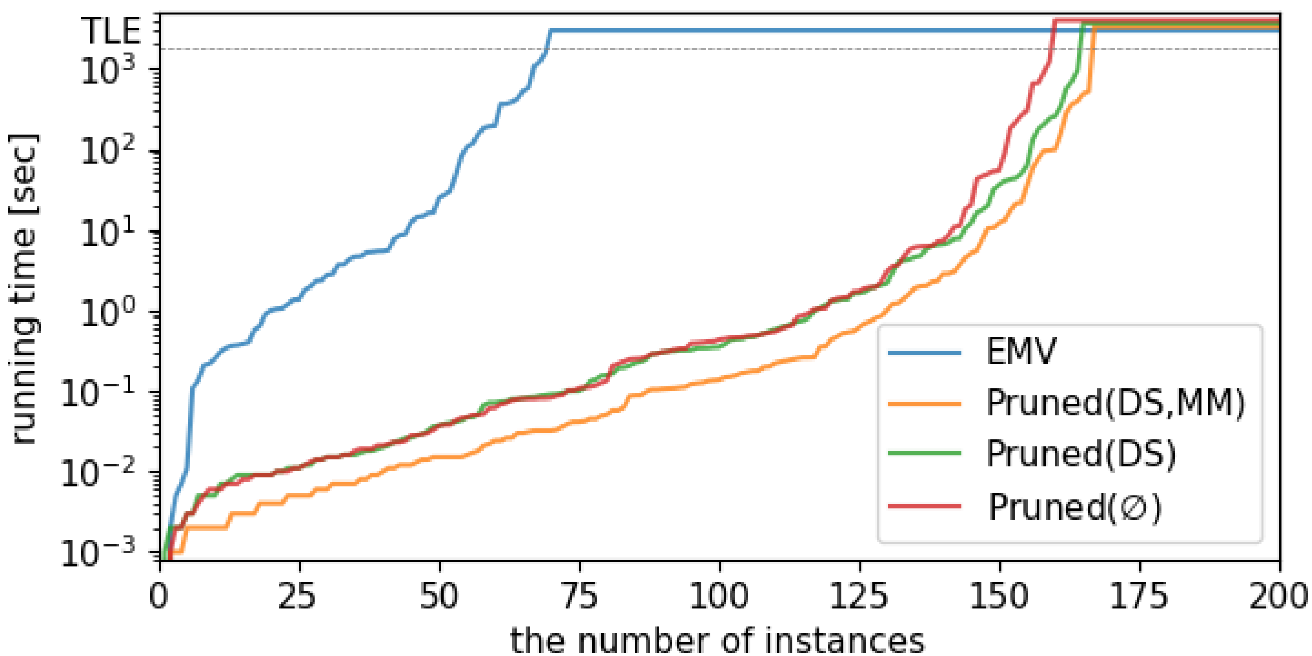
- ✓ No reductions
- ✓ 200 public instances from PACE



# Power of other techniques

DS: Data Structure

MM: Meet in the Middle



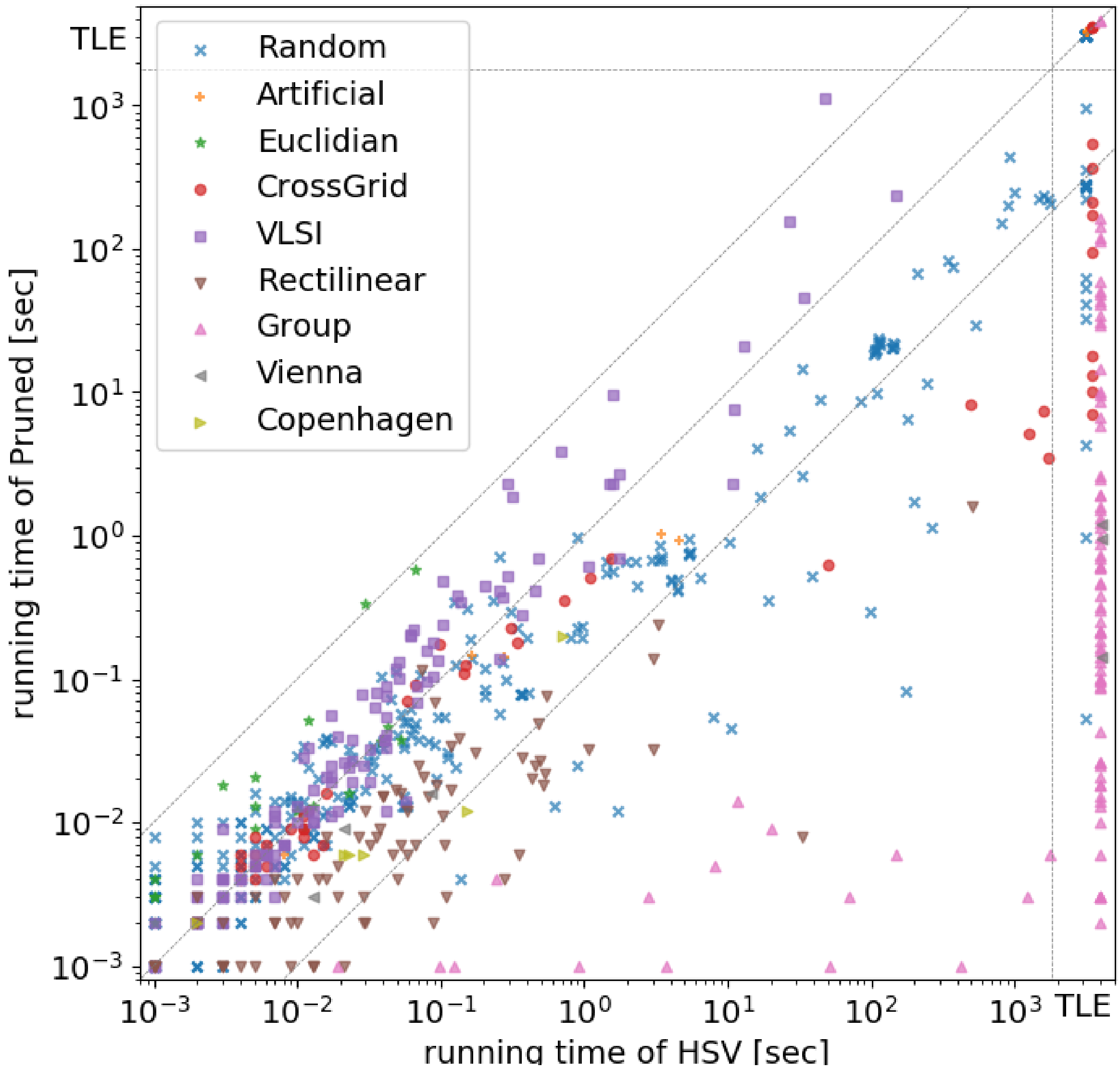
# Comparison with HSV

| Dataset | # | HSV solved | HSV better | PRUNED solved | PRUNED better |
|---|---|---|---|---|---|
| Random | 418 | 262 | 0 | **281** | **35** |
| Artificial | 21 | 11 | 0 | 11 | 0 |
| Euclidian | 26 | 26 | 0 | 26 | 0 |
| CrossGrid | 45 | 33 | 0 | **42** | **14** |
| VLSI | 128 | 128 | 1 | 128 | 0 |
| Rectilinear | 93 | 93 | 0 | 93 | 4 |
| Group | 89 | 16 | 0 | **87** | **82** |
| Vienna | 6 | 3 | 0 | **6** | **3** |
| Copenhagen | 10 | 10 | 0 | 10 | 0 |

✓ No reductions

$A$ is '**better**' than $B$ on an instance $i$ if
$A$ could solve $i$ but B couldn't or
(run-time of $B$) > (run-time of $A$) × 10 + 1s.

# Comparison with HSV



# Comparison with SCIP-Jack

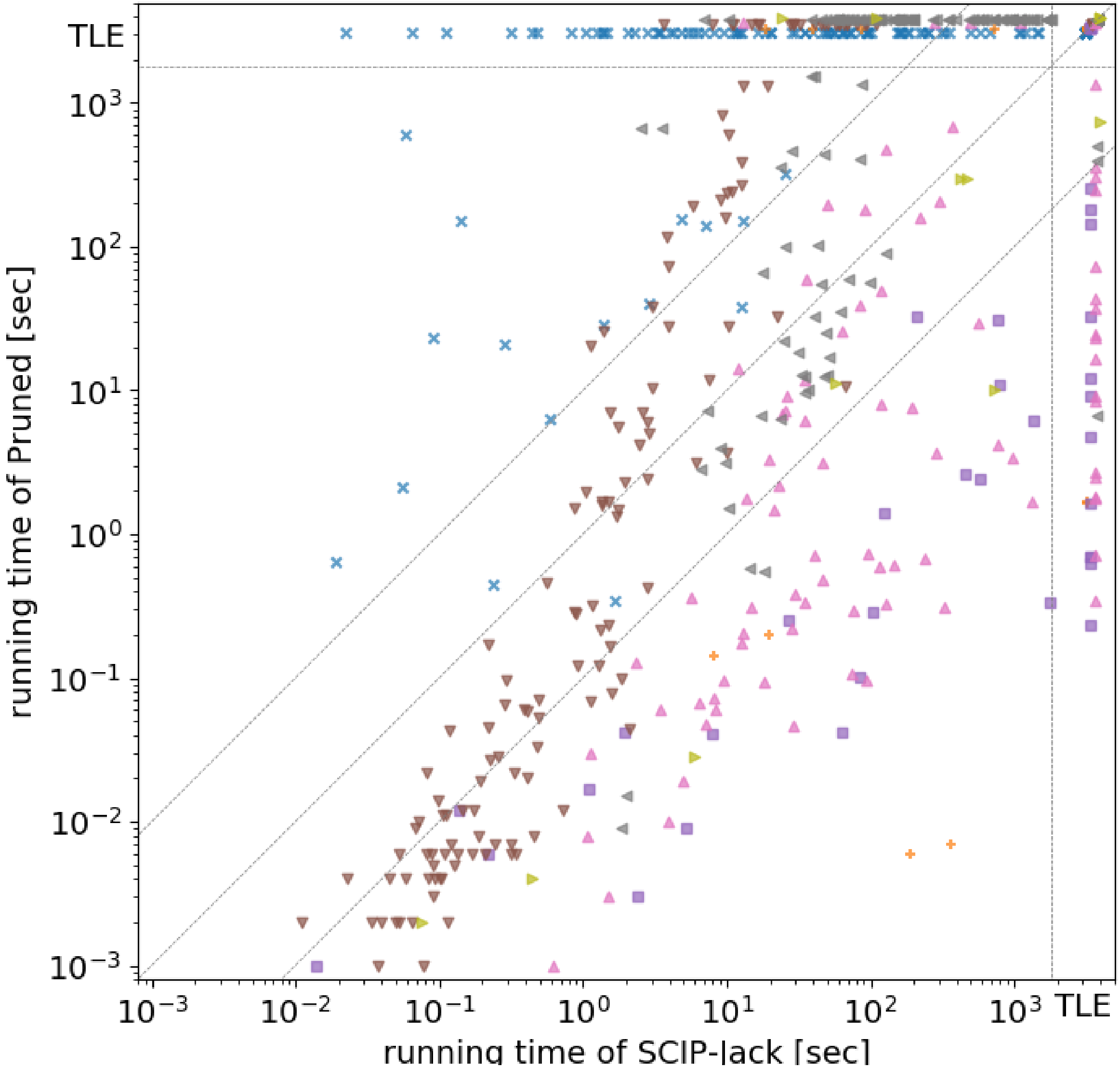
| Dataset | # | $k/n$ | SCIP-Jack solved | SCIP-Jack better | Pruned solved | Pruned better |
|---|---|---|---|---|---|---|
| Random | 133 | .256 | **96** | **91** | 16 | 0 |
| Artificial | 53 | .239 | 8 | 4 | 6 | 6 |
| VLSI | 40 | .046 | 20 | 0 | **32** | **27** |
| Rectilinear | 143 | .425 | **138** | **41** | 113 | 1 |
| Group | 85 | .086 | 65 | 4 | **79** | **58** |
| Vienna | 210 | .145 | **129** | **97** | 42 | 7 |
| Copenhagen | 12 | .140 | 8 | 2 | 7 | 3 |

average of $k/n$

✓ Pruned used the same reductions as SCIP-Jack

✓ Omit too-easy instances solved by the reductions alone

# Comparison with SCIP-Jack



# Conclusion and Open Problems

- ✓ Pruned DP is quite effective for Steiner Tree.
- ✓ Further speedup?
  - ➢ Pruning interferes with future pruning…
- ✓ Pruning for tree-decomposition DP?