Approximating the Held-Karp Bound for Metric TSP in Nearly Linear Time

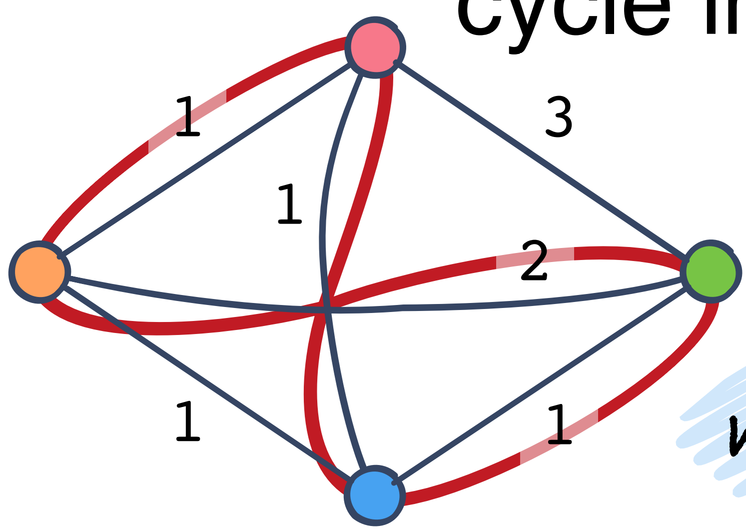
Chandra Chekuri

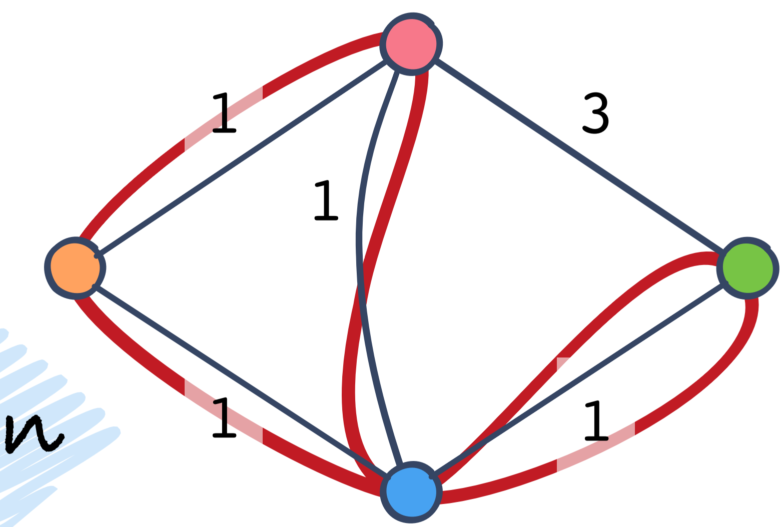
Kent Quanrud

UIUC

Metric TSP

version 1	***version 2***
Input: clique K_n , costs $c : V \times V \to \mathbb{R}_{\geq 0}$ forming a metric	**Input:** graph $G = (V, E)$ nonnegative costs $c \in \mathbb{R}^E$
Objective: min-cost Hamiltonian cycle in (K_n, c)	**Objective:** min-cost tour in (G, c)





Metric TSP

version 1

Input: clique K_n,

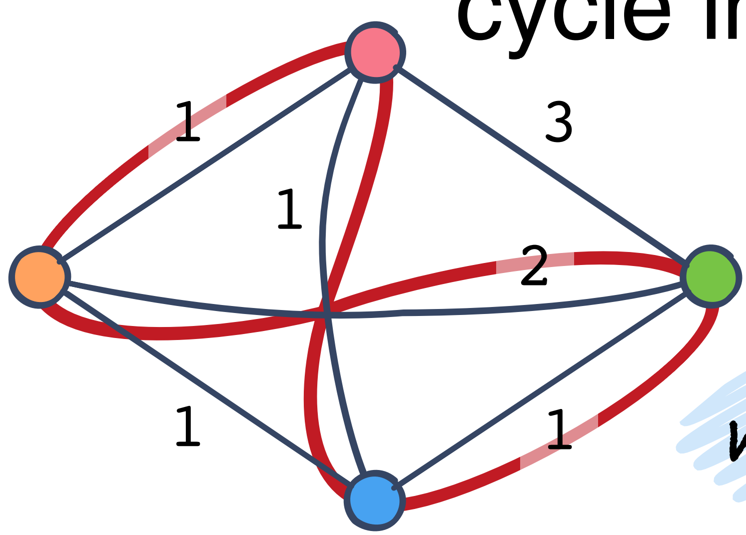
costs $\boxed{c : V \times V \to \mathbb{R}_{\geq 0}}$

forming a metric

$\boxed{(K_n, c) \text{ is dense (size } \Omega(n^2))}$

Objective:

min-cost Hamiltonian cycle in (K_n, c)



version 2

Input: graph $\boxed{G = (V, E)}$

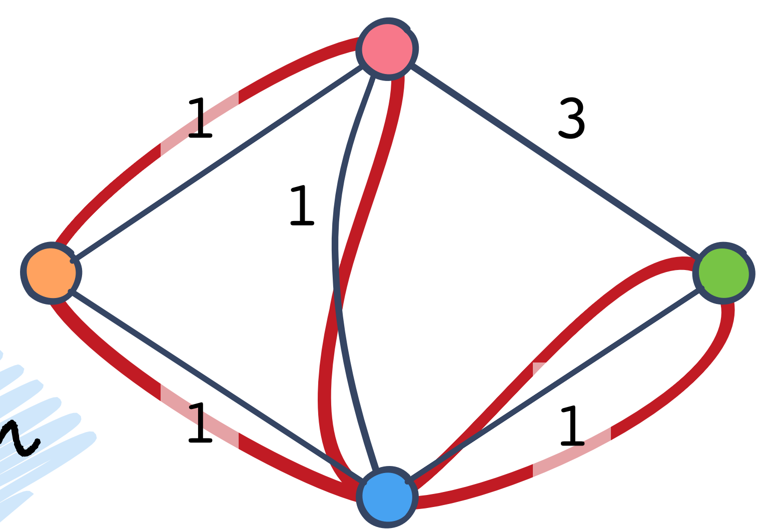
nonnegative costs $\boxed{c \in \mathbb{R}^E}$

$\boxed{(G, c) \text{ is sparse (size } O(m))}$

Objective:

min-cost tour in (G, c)

metric completion



(Metric) Subtour Elimination

Input: clique $K_n = (V, E)$, metric $c : E \to \mathbb{R}_{\geq 0}$

Objective: $\min \sum_{e \in E} c_e x_e$ over $x \in \mathbb{R}^E$

degree constraints

$$\text{s.t.} \sum_{e \in \mathscr{C}(v)} x_e = 2 \text{ for all vertices } v;$$

eliminates subtours

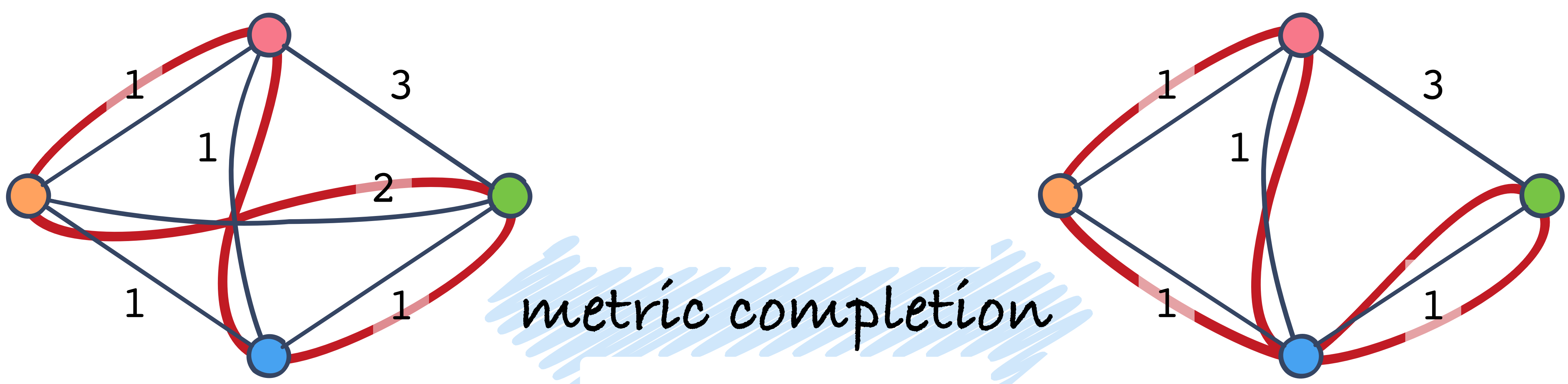
$$\sum_{e \in \mathscr{C}(U)} x_e \geq 2 \text{ for all sets } U \neq \emptyset, V;$$

$$\text{and } \mathbb{0} \leq x \leq \mathbb{1}$$

equivalent to Held-Karp bound

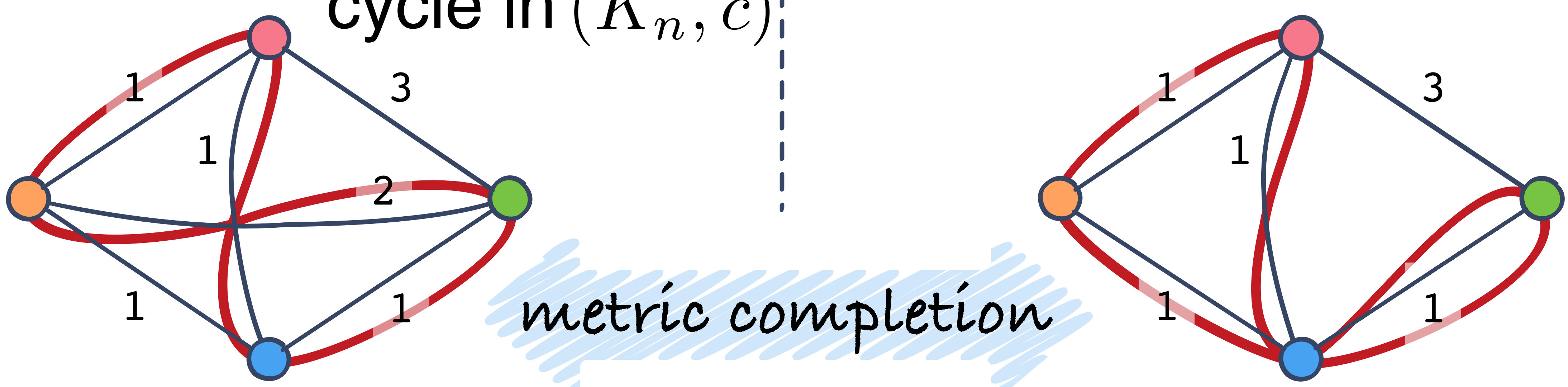
Related work (abbrev.)

- important for variants of Christofides algorithm
- solvable by ellipsoid method
- $\tilde{O}(n^4/\epsilon^2)$ *Plotkin, Shmoys, Tardos [1995]*
- $\tilde{O}(m^2/\epsilon^2)$ *Garg and Khandekar [2004]*



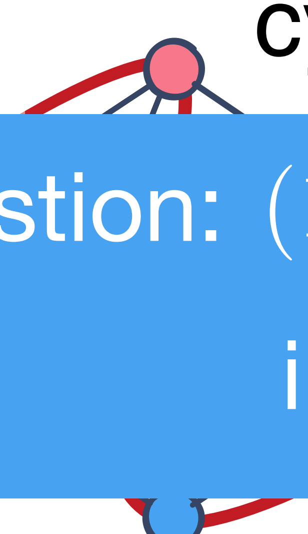
Metric TSP

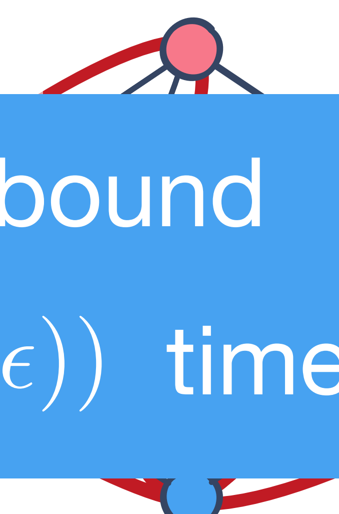
version 1	***version 2***
Input: clique K_n , costs $c : V \times V \to \mathbb{R}_{\geq 0}$ forming a metric	**Input:** graph $G = (V, E)$ nonnegative costs $c \in \mathbb{R}^E$
Objective: min-cost Hamiltonian cycle in (K_n, c)	**Objective:** min-cost tour in (G, c)



Metric TSP

version 1	*version 2*
Input: clique K_n , costs $c : V \times V \to \mathbb{R}_{\geq 0}$ forming a metric	**Input:** graph $G = (V, E)$ nonnegative costs $c \in \mathbb{R}^E$
Objective: min-cost Hamiltonian cycle in (K_n, c)	**Objective:** min-cost tour in (G, c)





Question: $(1-\epsilon)$ -APX for Held-Karp bound in nearly-linear $\tilde{O}(m/\mathrm{poly}(\epsilon))$ time?

Metric TSP

version 1

Input: clique K_n ,
costs $c : V \times V \to \mathbb{R}_{\geq 0}$
forming a metric

Objective:
min-cost Hamiltonian
cycle in (K_n, c)

version 2

Input: graph $G = (V, E)$
nonnegative costs $c \in \mathbb{R}^E$

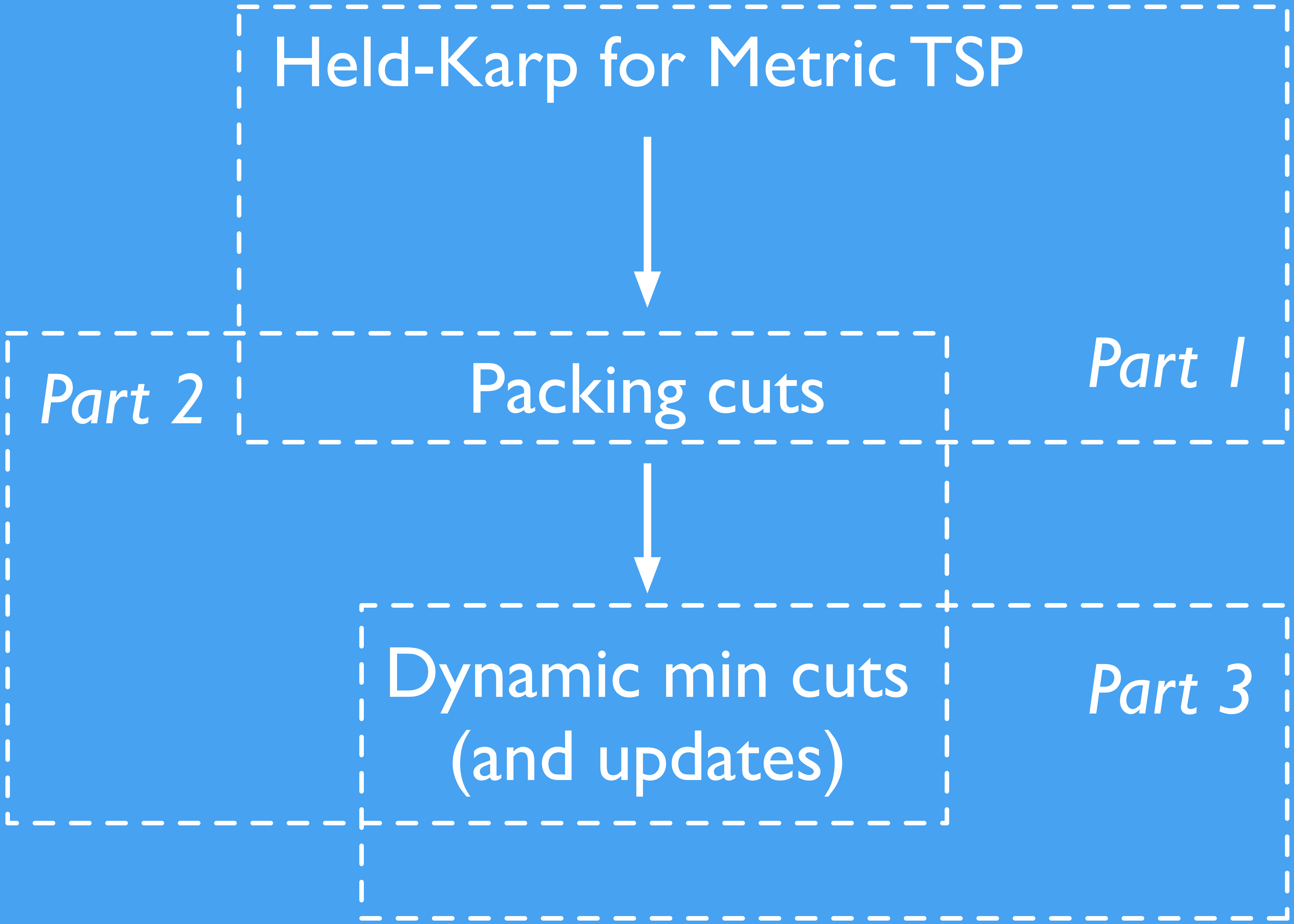
Objective:
min-cost tour in (G, c)

Yes: $(1 - \epsilon)$ -APX for Held-Karp in $\tilde{O}(m/\epsilon^2)$

Held-Karp for Metric TSP

Packing cuts

Dynamic min cuts
(and updates)



Held-Karp for Metric TSP
Packing cuts
Part 1
Part 2
Dynamic min cuts
(and updates)
Part 3

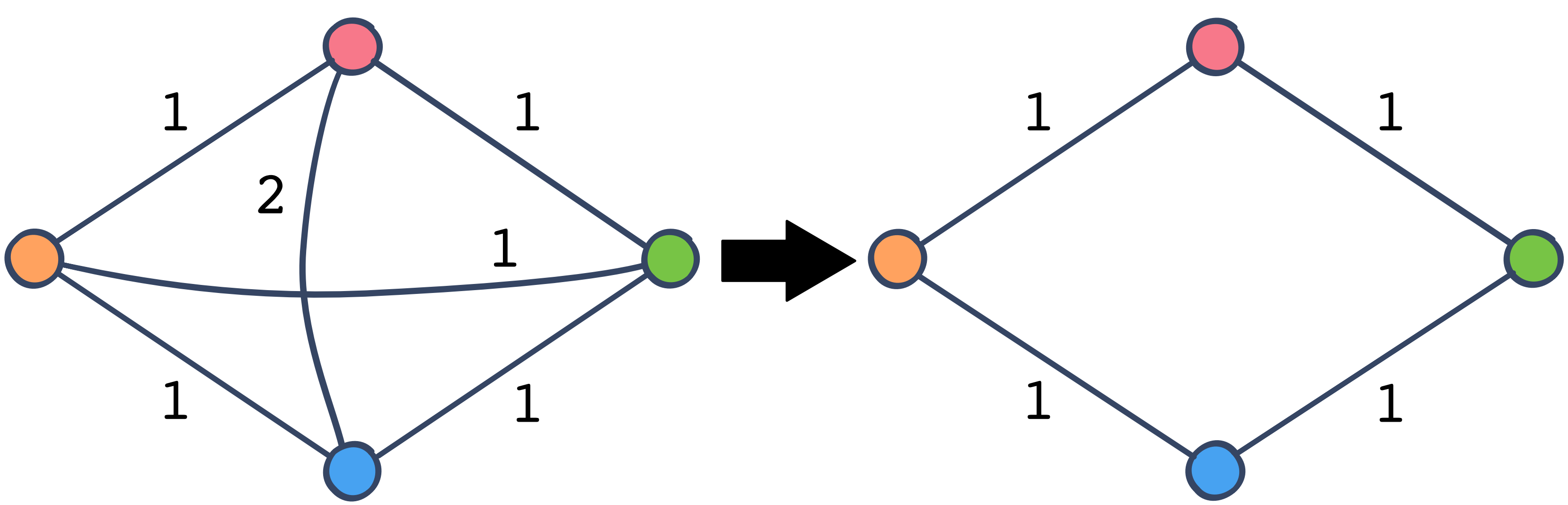
2-edge connected spanning subgraph (2ECSS)

Input: graph $G = (V, E)$ with cuts $\mathcal{C}$
nonnegative edge costs $c \in \mathbb{R}^E_{\geq 0}$

Objective: $\min \sum_{e \in E} c_e y_e$ over $y \in \mathbb{R}^E$

s.t. $\sum_{e \in \mathcal{C}} y_e \geq 2$ for all cuts $C \in \mathcal{C}$

and $y \geq \mathbb{0}^E$



equivalent to subtour elimination on metric completion of (G, c)

Dual of 2ECSS (2ECSSD)

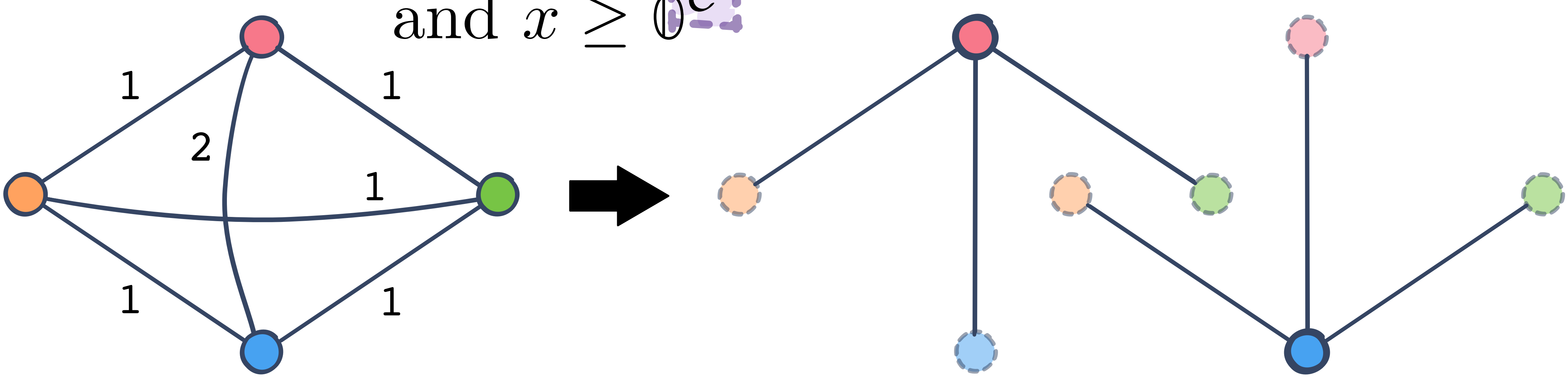
Input: graph $G = (V, E)$ with cuts $\mathcal{C}$, costs $c \in \mathbb{R}^{E}_{\geq 0}$

dimension $|\mathcal{C}|$ exponential in m

Objective: $$\max\ 2 \sum_{c \in \mathcal{C}} x_C \text{ over } x \in \mathbb{R}^{\mathcal{C}}$$

$$\text{s.t. } \sum_{C \ni e} x_C \leq c_e \text{ for all edges } e,$$

$$\text{and } x \geq \mathbb{0}^{\mathcal{C}}$$



Dual of 2ECSS (2ECSSD)

Input: graph $G = (V, E)$ with cuts $\mathcal{C}$, costs $c \in \mathbb{R}^{E}_{\geq 0}$

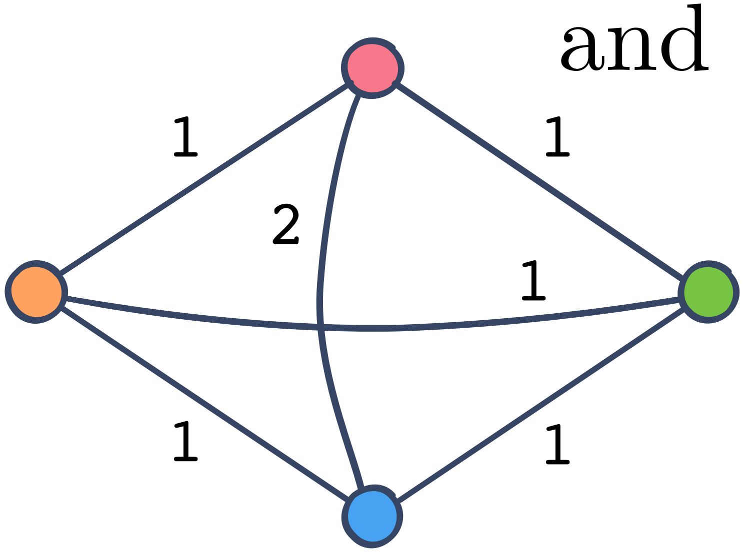
dimension $|\mathcal{C}|$ exponential in m

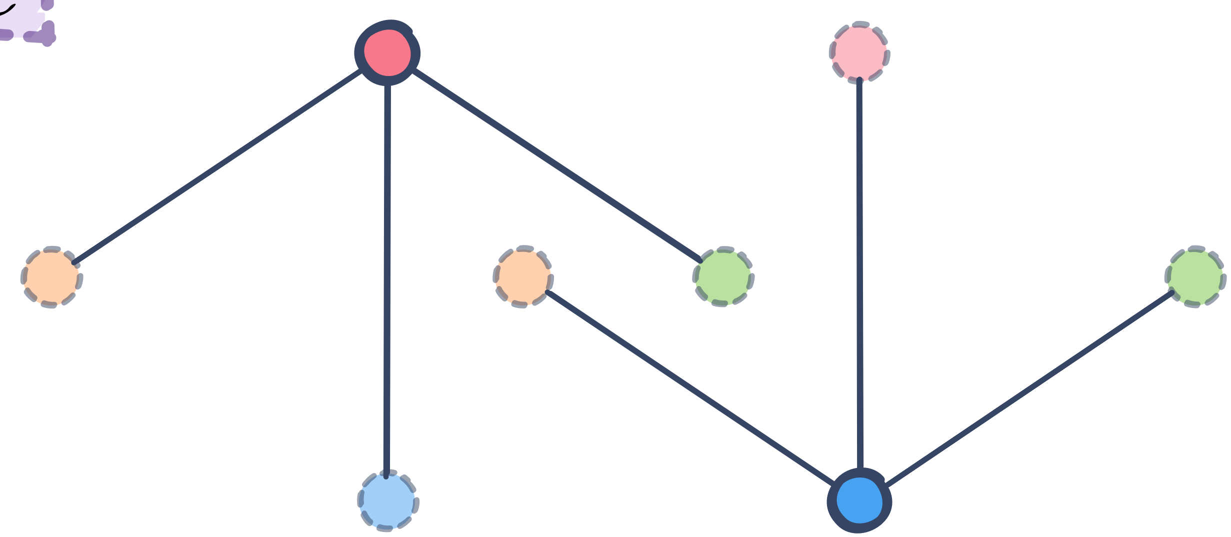
Objective: $\max 2 \sum_{c \in \mathcal{C}} x_C$ over $x \in \mathbb{R}^{\mathcal{C}}$

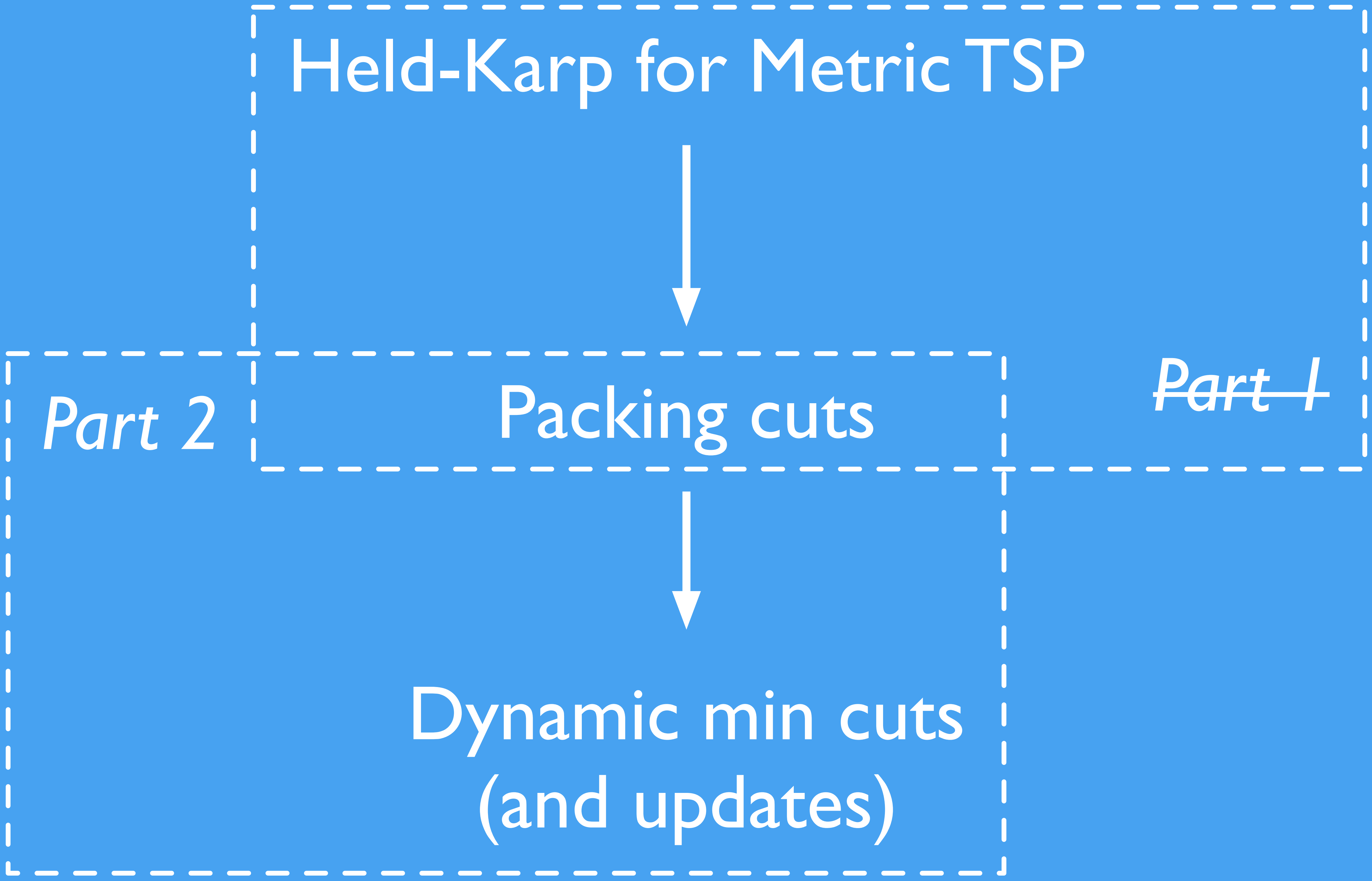
{0,1}-packing matrix

s.t. $\sum_{C \ni e} x_C \leq c_e$ for all edges e,

and $x \geq \mathbb{0}^{\mathcal{C}}$







Held-Karp for Metric TSP
Packing cuts
Part 2
~~Part 1~~
Dynamic min cuts
(and updates)

Multiplicative weight updates

cut packings $\xrightarrow{\texttt{MWU}}$ knapsack problems

$$\max \sum_{C \in \mathcal{C}} x_C \quad \text{over } x \in \mathbb{R}^{\mathcal{C}}$$

$$\text{s.t.} \sum_{C \ni e} x_C \leq c_e \qquad e \in E$$

$$x \geq \mathbb{0}^{\mathcal{C}}$$

0 `initialize edge weights` $w \leftarrow 1/c$

1 `solve relaxation`

$$\max \sum_{C} x_C \text{ s.t. } x \geq \mathbb{0},$$

$$\sum_{c,e:e \in c} w_e x_c \leq \sum_{e} w_e c_e$$

$\tilde{O}(m/\epsilon^2)$ `iterations`

2 `for each` $e \in E$

$$w_e \leftarrow \exp\left(\frac{\epsilon \sum_{C \ni e} \frac{x_C}{c_e}}{\max_h \sum_{C \ni h} \frac{x_C}{c_h}}\right) w_e$$

3 `output convex combination of relaxed solutions`

Multiplicative weight updates

cut packings $\xrightarrow{\texttt{MWU}}$ knapsack problems

$\boxed{1}$ `solve relaxation...`

$\boxed{0}$ `initialize edge weights` $w \leftarrow 1/c$

$\boxed{1}$ `solve relaxation`

$$\max \sum_C x_C \text{ s.t. } x \geq \mathbb{0},$$

$$\sum_{c,e:e\in c} w_e x_c \leq \sum_e w_e c_e$$

$\tilde{O}(m/\epsilon^2)$ `iterations`

$\boxed{2}$ `for each` $e \in E$

$$w_e \leftarrow \exp\left(\frac{\epsilon \sum_{C\ni e} \frac{x_C}{c_e}}{\max_h \sum_{C\ni h} \frac{x_C}{c_h}}\right) w_e$$

Multiplicative weight updates

cut packings $\xrightarrow{\texttt{MWU}}$ knapsack problems

1 solve relaxation...

a $C \leftarrow \text{min-cut}(G, w)$

$\tilde{O}(m)$ time [Karger]

b $x \leftarrow \frac{\langle w, c \rangle}{\sum_{e \in T} w_e} e_T$

0 initialize edge weights $w \leftarrow 1/c$

1 solve relaxation

$$\max \sum_C x_C \text{ s.t. } x \geq \mathbb{0},$$

$$\sum_{c,e:e \in c} w_e x_c \leq \sum_e w_e c_e$$

$\tilde{O}(m/\epsilon^2)$ iterations

2 for each $e \in E$

$$w_e \leftarrow \exp\left(\frac{\epsilon \sum_{C \ni e} \frac{x_C}{c_e}}{\max_h \sum_{C \ni h} \frac{x_C}{c_h}}\right) w_e$$

Multiplicative weight updates

cut packings → MWU → min-weight cuts

$\boxed{2}$ for each $e \in E$

$$w_e \leftarrow \exp\left(\frac{\epsilon \sum_{C \ni e} \frac{x_C}{c_e}}{\max_h \sum_{C \ni h} \frac{x_C}{c_h}}\right) w_e$$

$\boxed{0}$ initialize edge weights $w \leftarrow 1/c$

$\boxed{1}$ $\boxed{a}$ $C \leftarrow \text{min-cut}(G, w)$

$\tilde{O}(m)$ time [Karger]

$\boxed{b}$ $x \leftarrow \frac{\langle w, c\rangle}{\sum_{e \in T} w_e} e_T$

$\tilde{O}(m/\epsilon^2)$ iterations

$\boxed{2}$ for each $e \in E$

$$w_e \leftarrow \exp\left(\frac{\epsilon \sum_{C \ni e} \frac{x_C}{c_e}}{\max_h \sum_{C \ni h} \frac{x_C}{c_h}}\right) w_e$$

Multiplicative weight updates

cut packings $\xrightarrow{\texttt{MWU}}$ min-weight cuts

$\boxed{2}$ for each $e \in E$

$$w_e \leftarrow \exp\left(\frac{\epsilon \sum_{C \ni e} \frac{x_C}{c_e}}{\max_h \sum_{C \ni h} \frac{x_C}{c_h}}\right) w_e$$

$\Rightarrow$ for each $e \in C$,

$$w_e \leftarrow \exp\left(\frac{\epsilon \min_{h \in C} c_h}{c_e}\right) w_e$$

$\boxed{0}$ initialize edge weights $w \leftarrow 1/c$

$\boxed{1}$ $\boxed{a}$ $C \leftarrow \text{min-cut}(G, w)$

$\tilde{O}(m)$ time [Karger]

$\boxed{b}$ $x \leftarrow \frac{\langle w, c\rangle}{\sum_{e \in T} w_e} e_T$

$\tilde{O}(m/\epsilon^2)$ iterations

$\boxed{2}$ for each $e \in E$

$$w_e \leftarrow \exp\left(\frac{\epsilon \sum_{C \ni e} \frac{x_C}{c_e}}{\max_h \sum_{C \ni h} \frac{x_C}{c_h}}\right) w_e$$

Multiplicative weight updates

cut packings $\xrightarrow{\texttt{MWU}}$ min-weight cuts

$\boxed{0}$ `initialize edge weights` $w \leftarrow 1/c$

$\boxed{1}$ $\boxed{a}$ $C \leftarrow \text{min-cut}(G, w)$

$\tilde{O}(m)$ time [Karger]

$\boxed{b}$ $x \leftarrow \dfrac{\langle w, c\rangle}{\sum_{e \in T} w_e} e_T$

$\tilde{O}(m/\epsilon^2)$ `iterations`

$\boxed{2}$ `for` $e \in C$, $w_e \leftarrow \exp\left(\dfrac{\epsilon \min_{h \in C} c_h}{c_e}\right) w_e$

Multiplicative weight updates

cut packings → MWU → min-weight cuts

2 bottlenecks

0 `initialize edge weights` $w \leftarrow 1/c$

1 **a** $C \leftarrow \text{min-cut}(G, w)$

$\tilde{O}(m)$ time [Karger]

b $x \leftarrow \dfrac{\langle w, c\rangle}{\sum_{e \in T} w_e} e_T$

$\tilde{O}(m/\epsilon^2)$ `iterations`

2 `for` $e \in C,\ w_e \leftarrow \exp\left(\dfrac{\epsilon \min_{h \in C} c_h}{c_e}\right) w_e$

Multiplicative weight updates

cut packings $\xrightarrow{\texttt{MWU}}$ min-weight cuts

a $C \leftarrow \text{min-cut}(G, w)$

$\tilde{O}(m)$ per min cut

$\times \tilde{O}(m/\epsilon^2)$ iterations

✖ $\tilde{O}(m^2/\epsilon^2)$ running time

2 $\texttt{for } e \in C,\ w_e \leftarrow \cdots$

$\Omega(m)$ edge updates per cut

$\times \tilde{O}(m/\epsilon^2)$ iterations

✖ $\tilde{O}(m^2/\epsilon^2)$ running time

0 `initialize edge weights` $w \leftarrow 1/c$

1 **a** $C \leftarrow \text{min-cut}(G, w)$

$\tilde{O}(m)$ time [Karger]

b $x \leftarrow \dfrac{\langle w, c\rangle}{\sum_{e \in T} w_e} e_T$

$\tilde{O}(m/\epsilon^2)$ `iterations`

2 $\texttt{for } e \in C,\ w_e \leftarrow \exp\left(\dfrac{\epsilon \min_{h \in C} c_h}{c_e}\right) w_e$

From $\tilde{O}(m^2/\epsilon^2)$ to $\tilde{O}(m/\epsilon^2)$

	min-cut oracle	weight update
what we have	$\tilde{O}(m)$ per min cut	$\Omega(m)$ edges per cut
what we need	$\tilde{O}(1)$ amor. per $(1+\epsilon)$-apx min-cut	$\tilde{O}(1)$ amortized time per cut

additional challenges

- no suitable dynamic data structures
- min-cut varies dramatically between iterations

Held-Karp for Metric TSP

↓

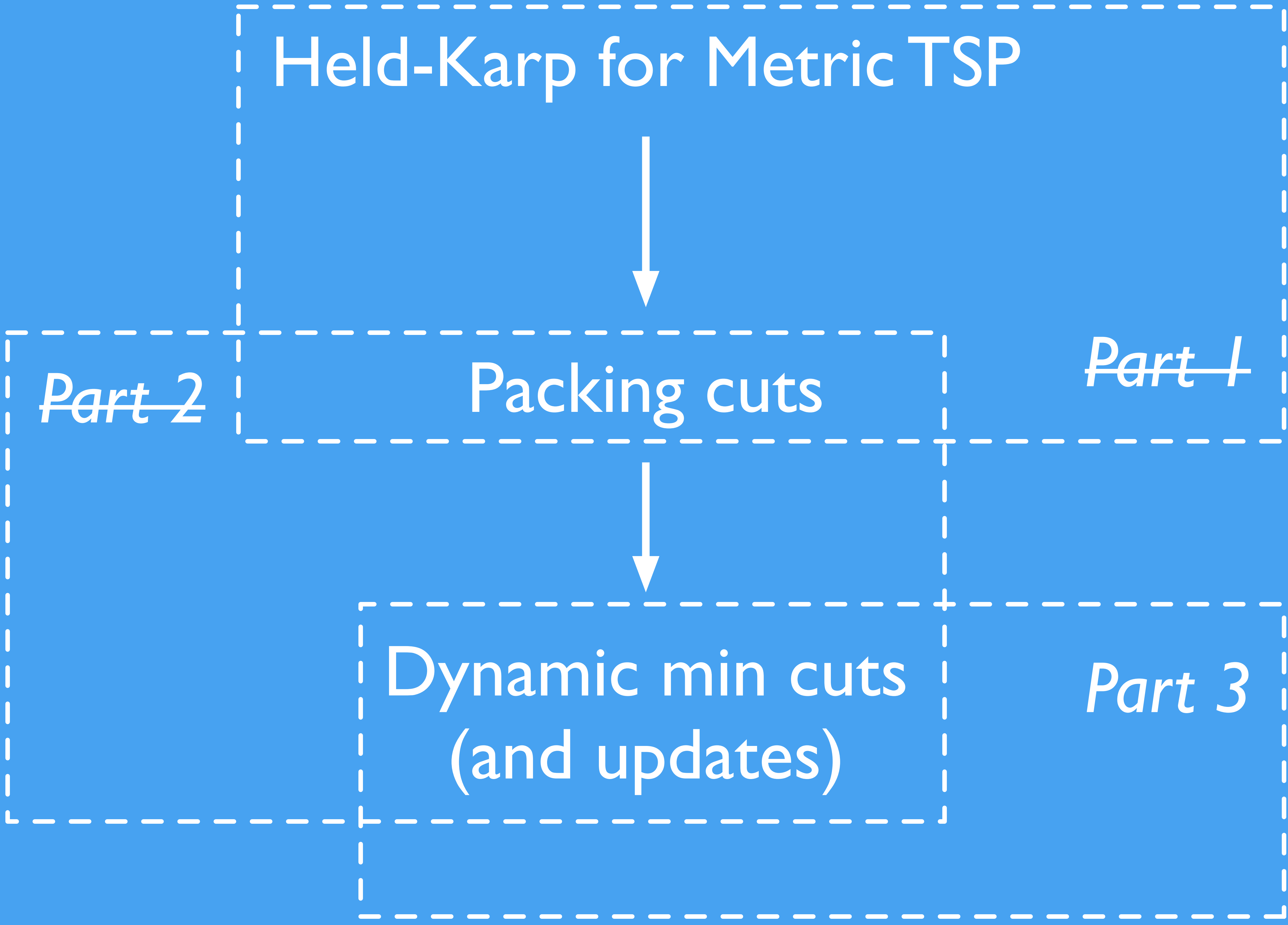
~~Part 2~~

Packing cuts

~~Part 1~~

↓

Dynamic min cuts
(and updates)


Held-Karp for Metric TSP
Packing cuts
Dynamic min cuts
(and updates)
~~Part 1~~
~~Part 2~~
Part 3

Karger's $\tilde{O}(m)$ min-cut algorithm

1. Randomly contract edge. Repeat.

Karger's $\tilde{O}(m)$ min-cut algorithm

~~1. Randomly contract edge. Repeat.~~

1. Pack spanning trees
2. Randomly sample $O(\log n)$ trees
3. Search each sampled tree for min-cut induced by 1 or 2 edges by dynamic programming

Tree packings and network strength

Input: graph $G = (V, E)$ w/ spanning trees $\mathcal{T}$
and positive edge capacities $c \in \mathbb{R}^E$

Objective: $$\max \sum_{T \in \mathcal{T}} x_t \text{ over } x \in \mathbb{R}^{\mathcal{T}}$$

$$\text{s.t.} \sum_{T \ni e} x_T \leq c_e \text{ for each edge } e$$

$$x \geq \mathbb{0}^{\mathcal{T}}$$

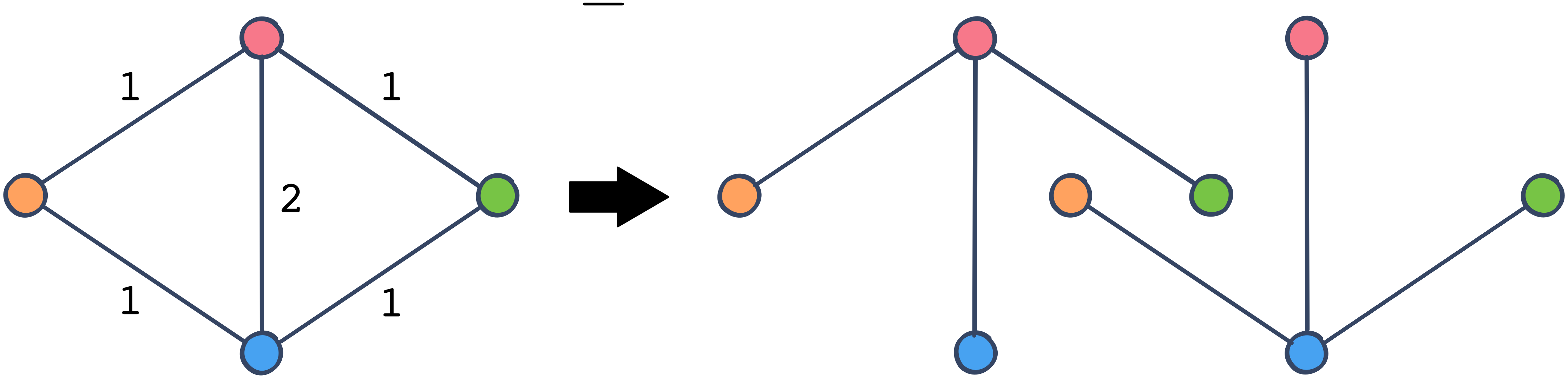
$(1-\epsilon)$-apx tree packings in $\tilde{O}(m)$ time via either:

- sparsification [Karger 00] - MWU [CQ SODA17]

Tree packings and network strength

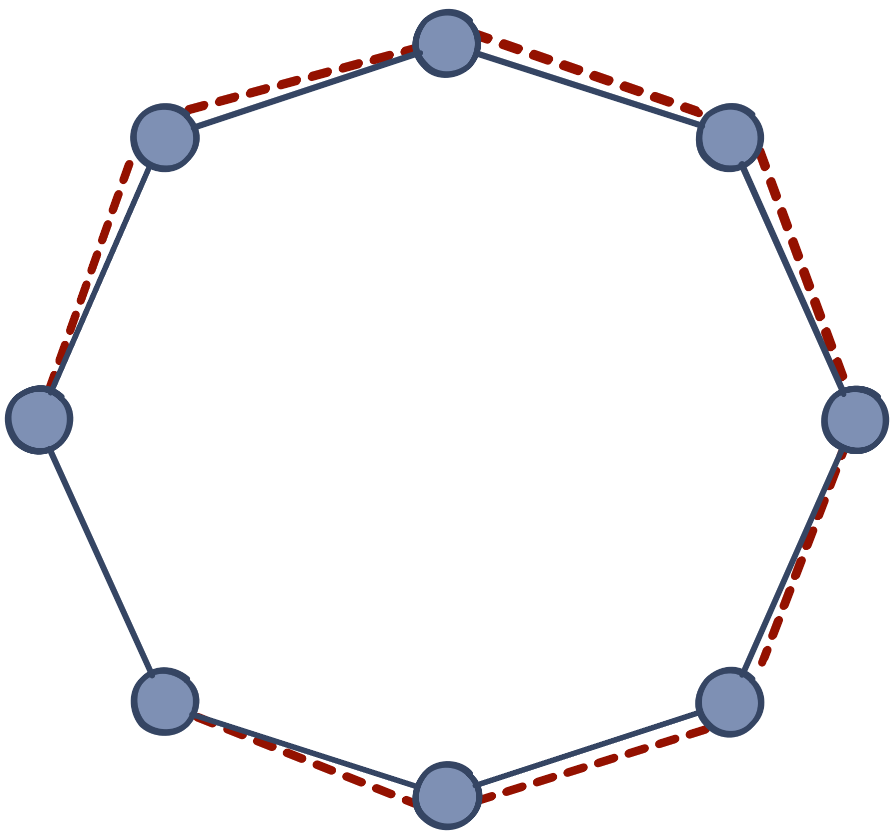
Input: graph $G = (V, E)$ w/ spanning trees $\mathcal{T}$ and positive edge capacities $c \in \mathbb{R}^E$

Objective:
$$\max \sum_{T \in \mathcal{T}} x_t \text{ over } x \in \mathbb{R}^{\mathcal{T}}$$
$$\text{s.t.} \sum_{T \ni e} x_T \leq c_e \text{ for each edge } e$$
$$x \geq \mathbb{0}^{\mathcal{T}}$$



Tutte **Nash-Williams** **1961**

Undirected graph w/ min cut κ has a tree packing of value $\geq \kappa/2$



Tight for cycle

- min cut = 2
- tree-packing = 1

Tutte **Nash-Williams** **1961**

Undirected graph w/ min cut κ has a tree packing of value $\geq \kappa/2$

0 let P be a $(1-\epsilon)$-apx max tree packing
C be a $(1+\epsilon)$-apx min cut

1 each edge $e \in C$ is in a tree $T \in P$

2 avg # C-edges per tree

$$= \frac{|C|}{|P|} \leq \frac{(1+\epsilon)\kappa}{(1-\epsilon)\kappa/2} = 2 + O(\epsilon)$$

3 **Markov $\Rightarrow$ const. fraction of trees have < 3 C-edges**

Karger's $\tilde{O}(m)$ min-cut algorithm

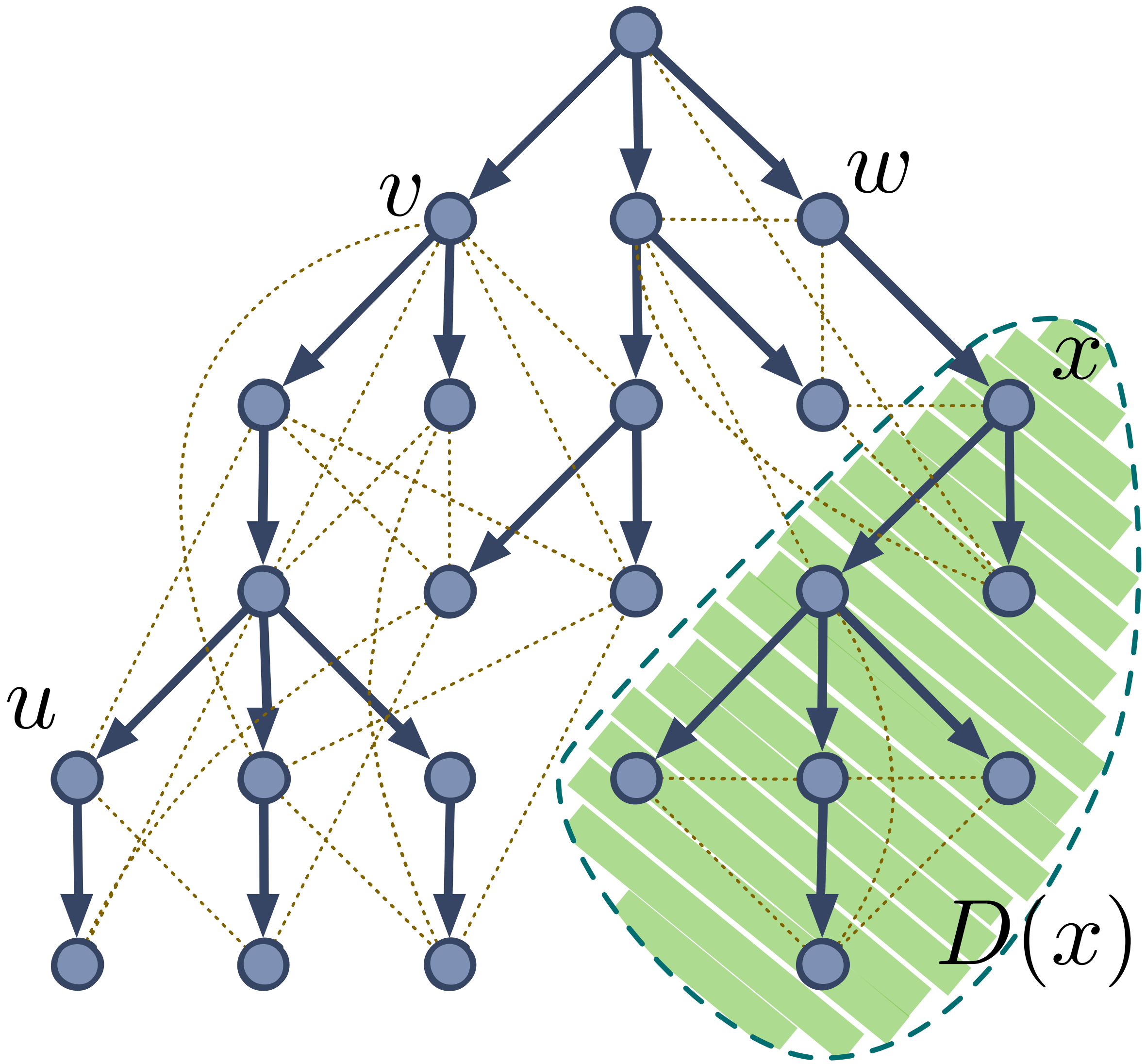
~~1. Randomly contract edge. Repeat.~~

1. Pack spanning trees

2. Randomly sample $O(\log n)$ trees

3. Search each sampled tree for min-cut induced by 1 or 2 edges by dynamic programming

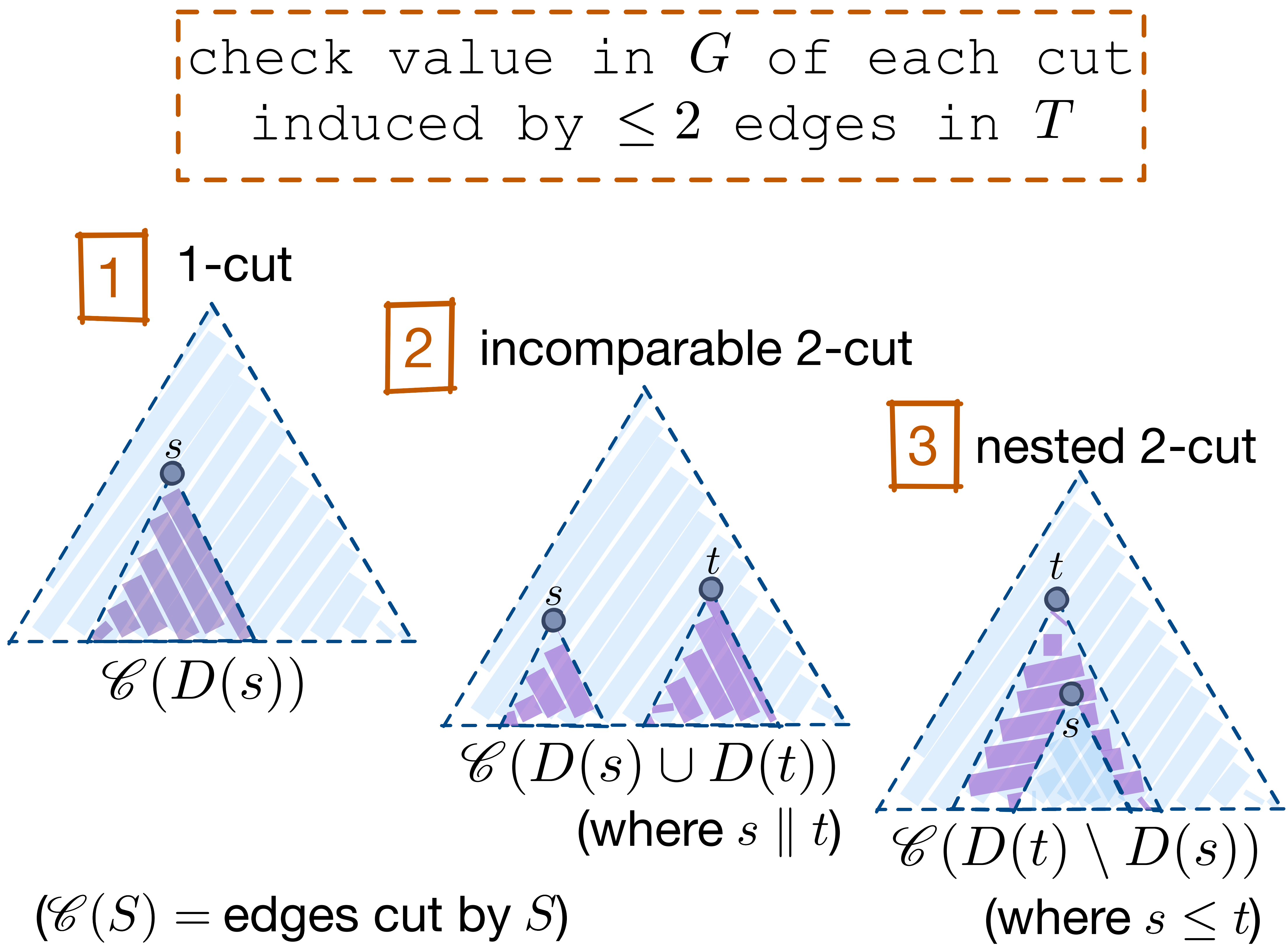
Fix a rooted tree $T \Rightarrow$ poset on V



"$u < v$" means u is a descendant of v

"$v \parallel w$" means v and w are incomparable

"$D(x)$" means all descendants of x



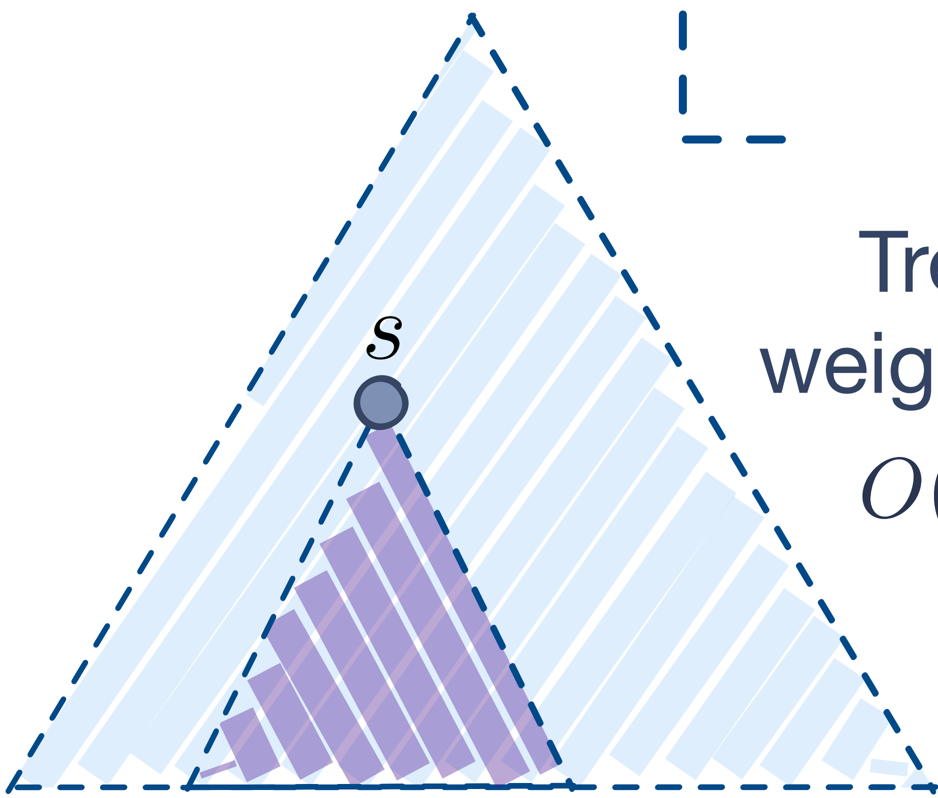
check value in G of each cut
induced by ≤ 2 edges in T
1
1-cut
s
$\mathscr{C}(D(s))$
2
incomparable 2-cut
s
t
$\mathscr{C}(D(s) \cup D(t))$
(where $s \parallel t$)
3
nested 2-cut
t
s
$\mathscr{C}(D(t) \setminus D(s))$
(where $s \leq t$)
($\mathscr{C}(S)$ = edges cut by S)

$[\overline{\mathscr{C}}(X) = \overline{w}(\mathscr{C}(X))]$

$\boxed{1}$ 1-cut

sum of weighted degrees

$$\underbrace{\overline{w}(\mathscr{C}(D(s)))}_{\text{weight of cut}} = \sum_{v\in D(s)} \underbrace{\sum_{e\in\mathscr{C}(v)} w(e)}_{\text{weighted degree}} - \underbrace{\sum_{e\in E[D(s)]} w(e)}_{\text{weight of edges contained in } D(s)}$$



$\mathscr{C}(D(s))$

Tree sums over weighted degrees

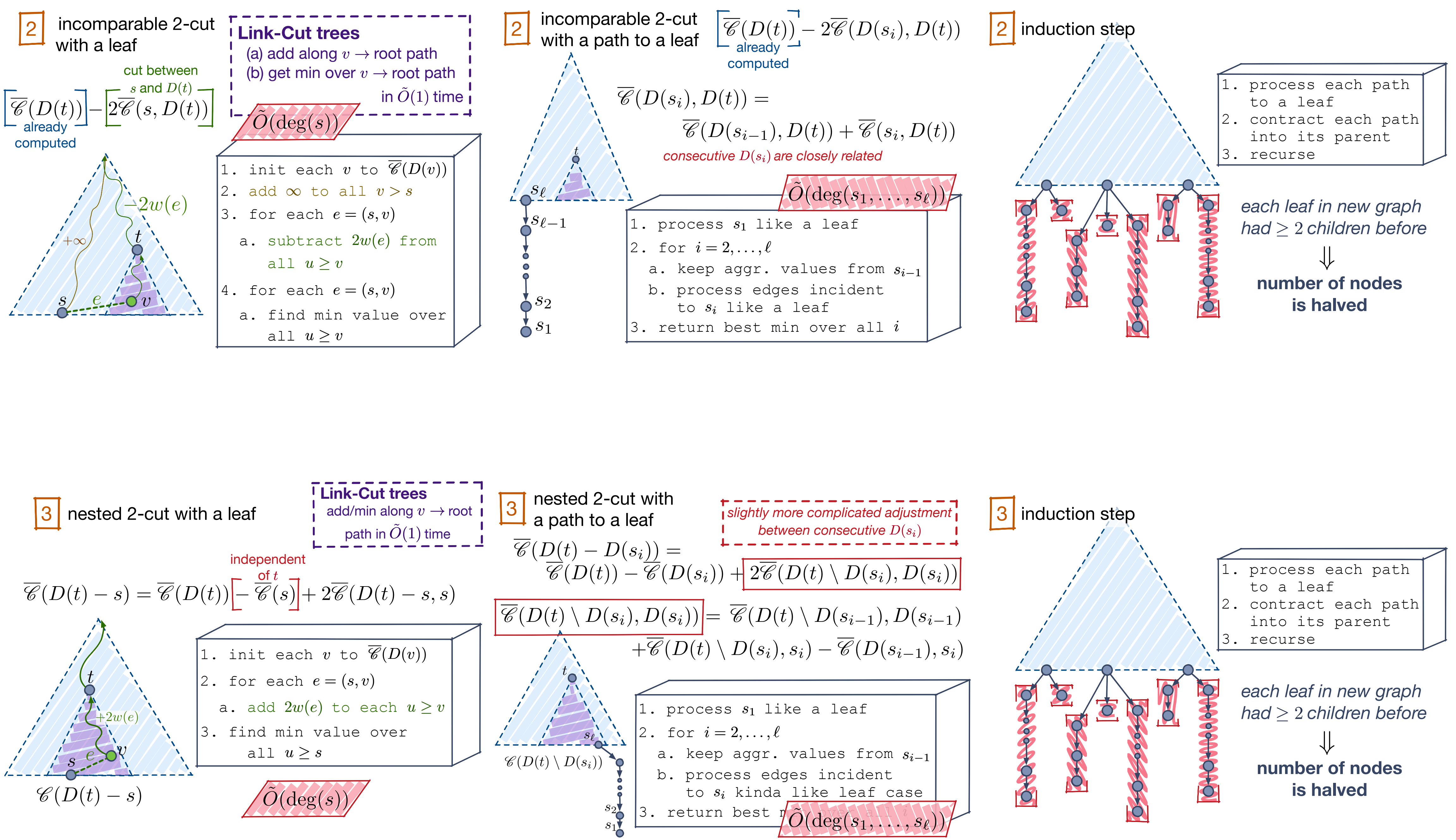
$O(m)$ time total over all s

Tree sums over weights at lca's

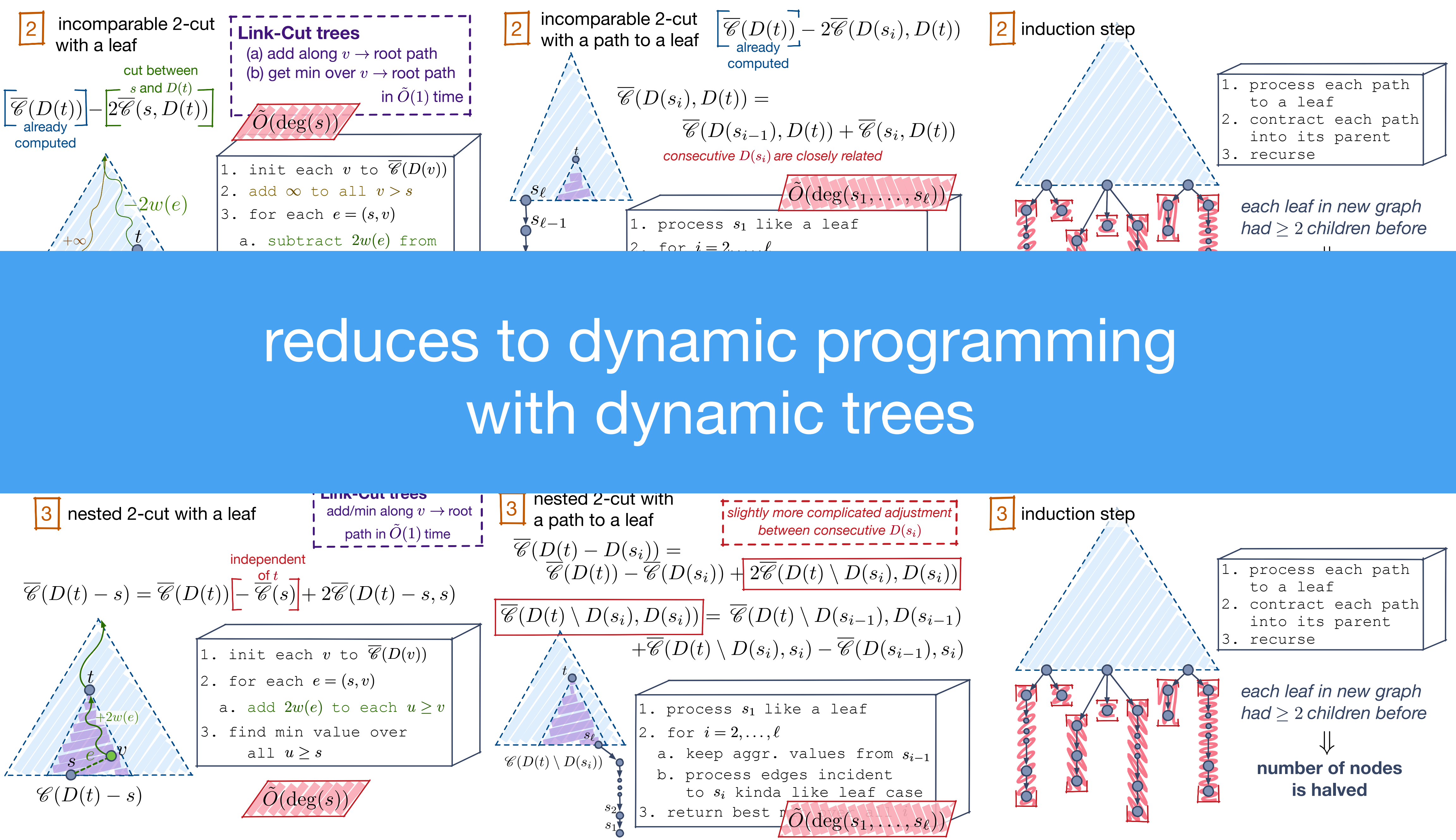
$\tilde{O}(m)$ time total over all s

build up sums in dynamic trees and read off weights

2-cuts are a little more complicated…



2-cuts are a little more complicated…



reduces to dynamic programming with dynamic trees

Incremental setting

- edge weights incremented online (adversarially)
- need to maintain $(1+\epsilon)$-apx min cut

Incremental Karger's algorithm

- initially: $\lambda \leftarrow$ initial value of min-cut,
  pack and sample $\log n$ spanning trees
- when we need an apx min-cut:
  - continue Karger's search until we find a cut of value $\leq (1+\epsilon)\lambda$, output and pause the search
  - if good cut not found, then re-pack/sample trees
- when we increment an edge weight
  - incorporate into tree sums w/ dynamic trees

From $\tilde{O}(m^2/\epsilon^2)$ to $\tilde{O}(m/\epsilon^2)$

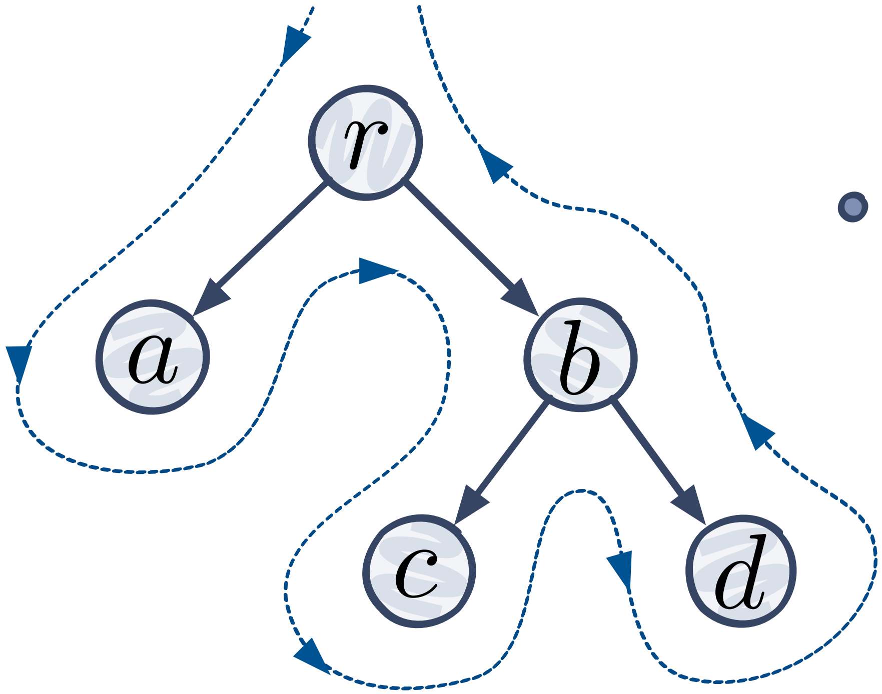
	min-cut oracle	weight update
what we have	$\tilde{O}(m)$ per min cut	$\Omega(m)$ edges per cut
what we need	$\tilde{O}(1)$ amor. per $(1+\epsilon)$-apx min-cut	$\tilde{O}(1)$ amortized time per cut
what we get	$\tilde{O}(m/\epsilon^2)$ total time $+\tilde{O}(1)$ per min cut $+\tilde{O}(1)$ per edge inc	??

Updating edge weights along cuts

- need to update weights of all edges in a cut
- we know how to update fixed sets efficiently

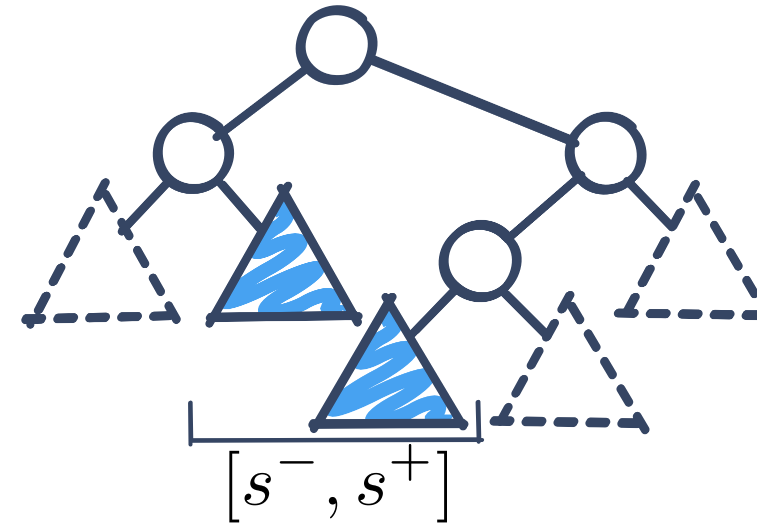
[Young '14, Chekuri-Q SODA17]

- problem: cuts vary dramatically between iterations
- key point: all cuts are induced by trees

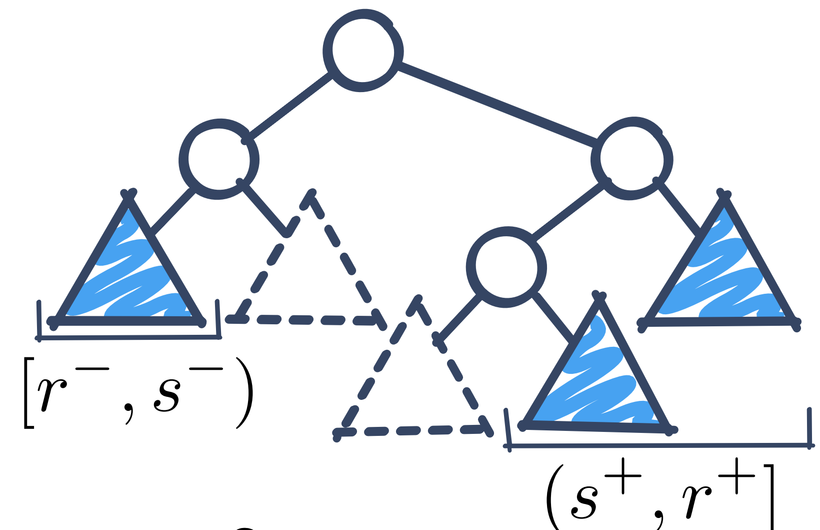


$r^- < a^- < a^+ < b^- < c^- <$
$c^+ < d^- < d^+ < c^+ < r^+$

- Euler tour converts subtrees to intervals



- Range tree decomposes each side of a 1,2-cut to $\log n$ “canonical subtrees”



- this decomposes each 1,2-cut to $\log^2 n$ “canonical cuts” between canonical subtrees

Updating edge weights along cuts

- need to update weights of all edges in a cut

★ we know how to update fixed sets efficiently

[Young '14, Chekuri-Q SODA17]

- problem: cuts vary dramatically between iterations

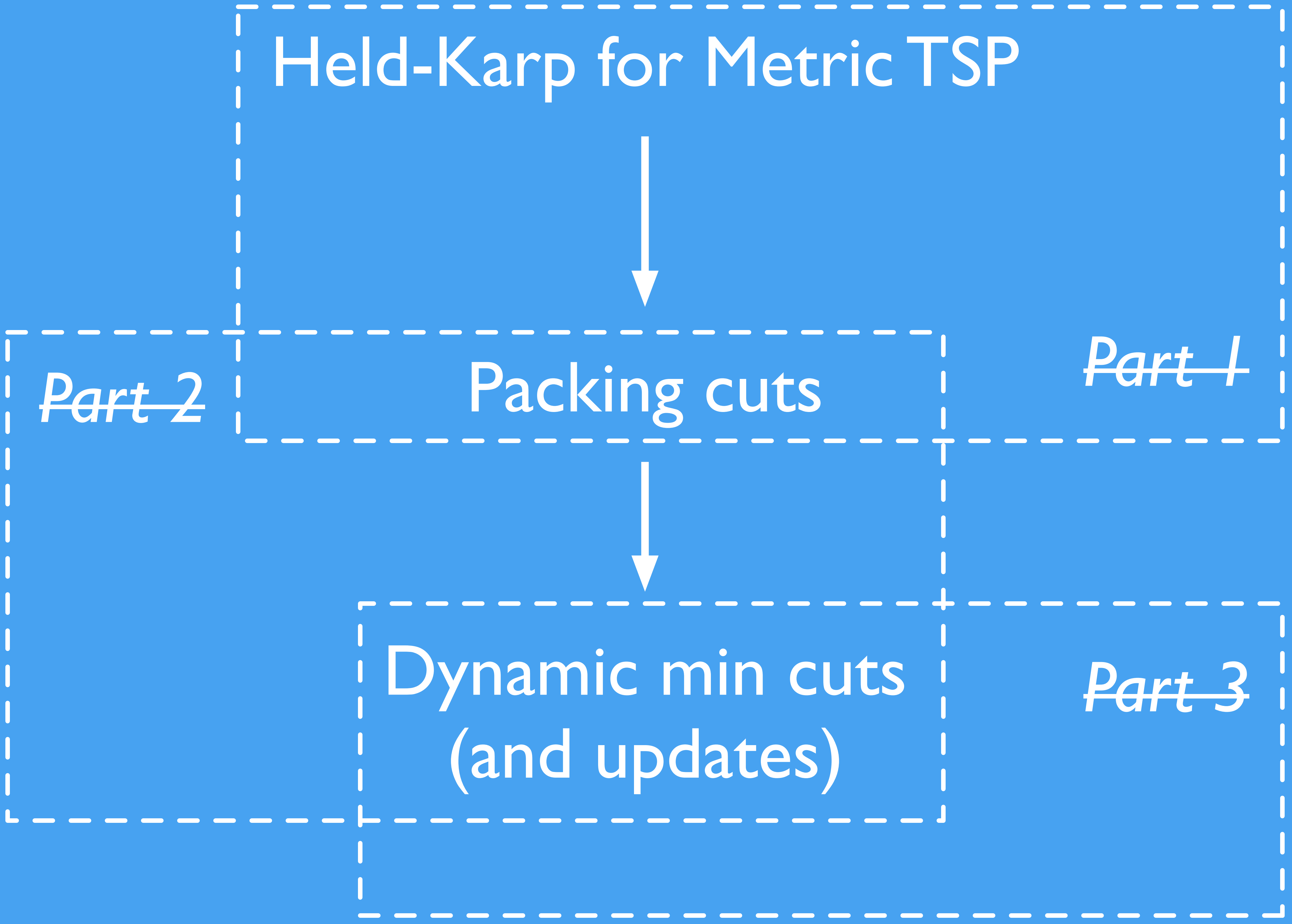
- key point: all cuts are induced by trees

  ★ "canonical cuts" with total size $\tilde{O}(m)$

★ + ★ => $\log^2 n$ efficient updates on fixed sets

From $\tilde{O}(m^2/\epsilon^2)$ to $\tilde{O}(m/\epsilon^2)$

	min-cut oracle	weight update
what we have	$\tilde{O}(m)$ per min cut	$\Omega(m)$ edges per cut
what we need	$\tilde{O}(1)$ amor. per $(1+\epsilon)$-apx min-cut	$\tilde{O}(1)$ amortized time per cut
what we get	$\tilde{O}(m/\epsilon^2)$ total time $+\tilde{O}(1)$ per min cut $+\tilde{O}(1)$ per edge inc	$\tilde{O}(m/\epsilon^2)$ init time $+\tilde{O}(1)$ per min cut $+\tilde{O}(m/\epsilon^2)$ edge increments

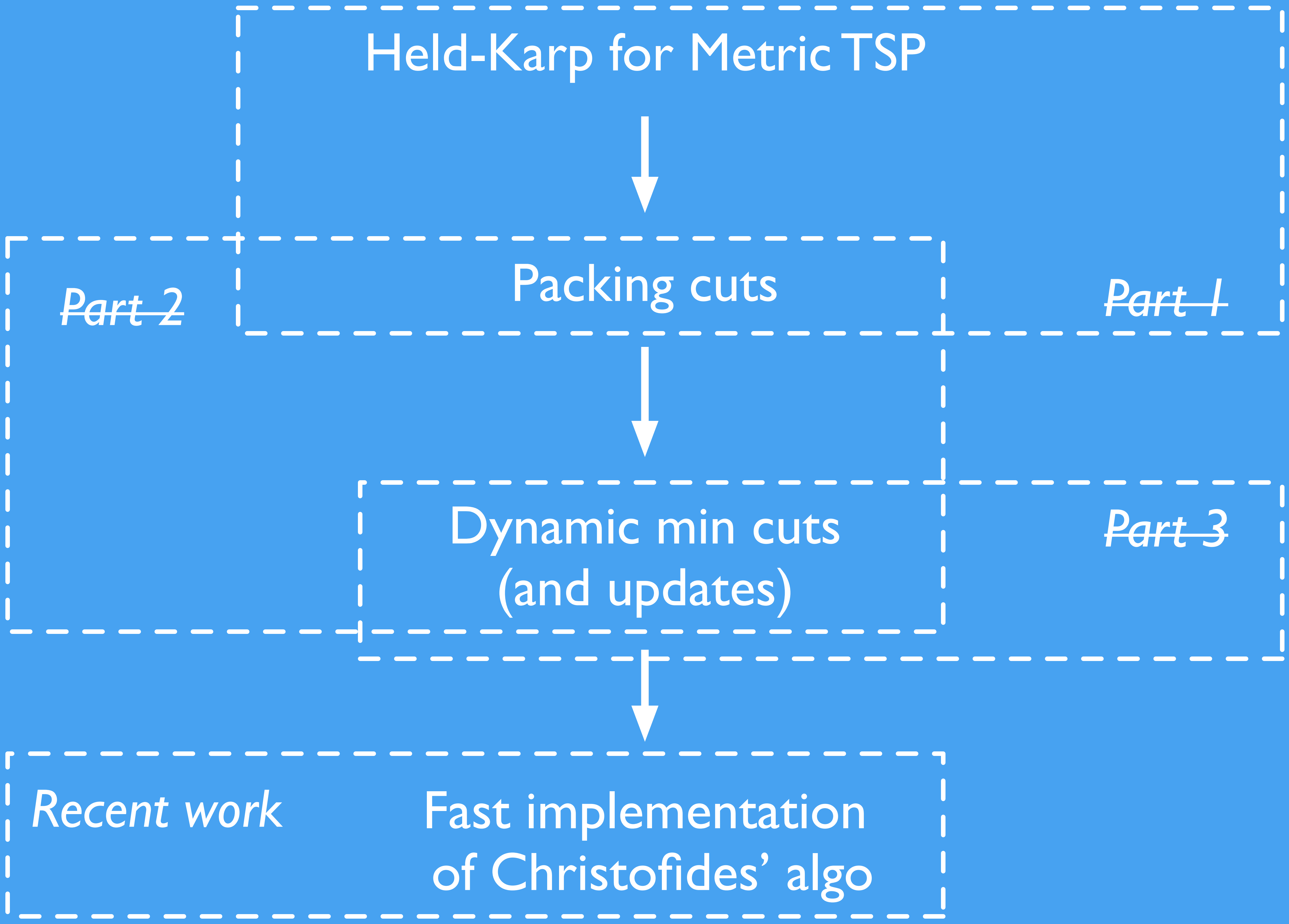

Held-Karp for Metric TSP
~~Part 1~~
~~Part 2~~
Packing cuts
Dynamic min cuts
(and updates)
~~Part 3~~

The main result. In this paper we obtain a near-linear running time for a $(1+\epsilon)$-approximation, substantially improving the best previously known running time bound.

Theorem 1.1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with $|\mathcal{E}| = m$ edges and $|\mathcal{V}| = n$ vertices, and positive edge weights $c : \mathcal{E} \to \mathbb{R}_{>0}$. For any fixed $\epsilon > 0$, there exists a randomized algorithm that computes a $(1+\epsilon)$-approximation to the Held-Karp lower bound for the* Metric-TSP *instance on $(\mathcal{G}, c)$ in $O(m \log^4 n/\epsilon^2)$ time. The algorithm succeeds with high probability.*

The algorithm in the preceding theorem can be modified to return a $(1+\epsilon)$-approximate solution to the 2ECSS LP within the same asymptotic time bound. For fixed ϵ, the running time we achieve is asymptotically faster than the time to compute or even write down the metric completion of $(\mathcal{G}, c)$. Our algorithm can be applied low-dimensional geometric point sets to obtain a running-time that is near-linearly in the number of points.

In typical approximation algorithms that rely on mathematical programming relaxations, the bottleneck for the running time is solving the relaxation. Surprisingly, for algorithms solving Metric-TSP via the Held-Karp bound, the bottleneck is no longer solving the relaxation (albeit we only find a $(1+\epsilon)$-approximation and do not guarantee a basic feasible solution). We mention that the recent approaches towards the 4/3 conjecture for Metric-TSP are based on variations of the classical Christofides heuristic (see [Vygen, 2012]). The starting point is a near-optimal feasible solution x to the 2ECSS LP on $(\mathcal{G}, c)$. Using a well-known fact that a scaled version of x lies in the spanning tree polytope of $\mathcal{G}$, one generates one or more (random) spanning trees T of $\mathcal{G}$. The tree T is then augmented to a tour via a min-cost matching M on its odd degree nodes. Genova and Williamson [2017] recently evaluated some of these *Best-of-Many Christofides' algorithms* and demonstrated their effectiveness. A key step in this scheme, apart from solving the LP, is to decompose a given


Held-Karp for Metric TSP
Packing cuts
~~Part 1~~
~~Part 2~~
Dynamic min cuts
(and updates)
~~Part 3~~
Recent work
Fast implementation
of Christofides' algo

Christofides' heuristic [1976]
(Recent work)

- simple (& best) 3/2-approximation for metric TSP
- bottlenecks include all-pairs shortest paths, min-cost perfect matching on dense graph
- $(1+\epsilon)$-apx to 2ECSS =>

  $(1+\epsilon)\frac{3}{2}$-apx in $\tilde{O}(n^{1.5}/\epsilon^3)$ time
- => $\tilde{O}(m/\epsilon^2 + n^{1.5}/\epsilon^3)$ time total

Thanks!