Approximating the Held-Karp Bound for Metric TSP in Nearly Linear Work and Polylog Depth

Zhuan Khye (Cedric) Koh

Omri Weinstein   Sorrachai Yingchareonthawornchai

BOSTON UNIVERSITY

האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

ETH zürich

Work-Depth Model

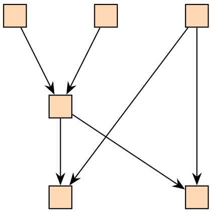
Work-Depth Model

Sequential computation

Work-Depth Model

Sequential computation

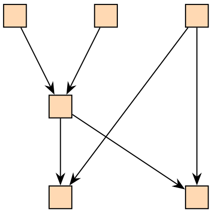
Parallel computation



Work-Depth Model

Sequential computation

Parallel computation

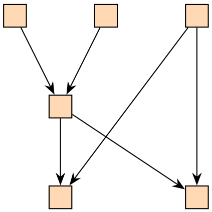


- Work = Total number of operations

Work-Depth Model

Sequential computation

Parallel computation

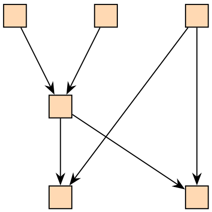


- Work = Total number of operations
- Depth = Length of a longest chain of dependent operations

Work-Depth Model

Sequential computation

Parallel computation



- Work = Total number of operations
- Depth = Length of a longest chain of dependent operations
- Fast parallel algorithm = Nearly linear work and polylog depth.

Metric TSP

- Given an undirected graph $G = (V, E)$ with edge costs $c \in \mathbb{R}^m_{\geq 0}$,

TSP: Find a minimum-cost Hamiltonian cycle in G.

Metric TSP

- Given an undirected graph $G = (V, E)$ with edge costs $c \in \mathbb{R}^m_{\geq 0}$,

TSP: Find a minimum-cost Hamiltonian cycle in G.

- Inapproximable

Metric TSP

- Given an undirected graph $G = (V, E)$ with edge costs $c \in \mathbb{R}^m_{\geq 0}$,

TSP: Find a minimum-cost Hamiltonian cycle in G.

- Inapproximable

Metric TSP: Find a minimum-cost spanning tour in G.

Metric TSP

- Given an undirected graph $G = (V, E)$ with edge costs $c \in \mathbb{R}^m_{\geq 0}$,

TSP: Find a minimum-cost Hamiltonian cycle in G.

- Inapproximable

Metric TSP: Find a minimum-cost spanning tour in G.

- **APX**-hard [Lampis '14]
- $3/2$ approximation [Christofides '76] [Serdyukov '78]
- $3/2 - 10^{-36}$ approximation [Karlin, Klein, Oveis Gharan '22]

Metric TSP

- Given an undirected graph $G = (V, E)$ with edge costs $c \in \mathbb{R}^m_{\geq 0}$,

TSP: Find a minimum-cost Hamiltonian cycle in G.

- Inapproximable

Metric TSP: Find a minimum-cost spanning tour in G.

- **APX**-hard [Lampis '14]
- $3/2$ approximation [Christofides '76] [Serdyukov '78]
- $3/2 - 10^{-36}$ approximation [Karlin, Klein, Oveis Gharan '22]

- Metric TSP on $(G, c) \equiv$ TSP on the metric completion $(\hat{G}, \hat{c})$

$$\hat{G} = \text{Complete graph on } V$$
$$\hat{c}_{uv} = \text{Shortest path length between } u \text{ and } v \text{ in } G$$

Subtour Elimination LP

[Dantzig, Fulkerson, Johnson '54]

$$
\begin{aligned}
\min\ & \hat{c}^\top x \\
\text{s.t.}\ & \sum_{v} x_{uv} = 2 && \forall\, u \in V \\
& \sum_{u \in S, v \notin S} x_{uv} \geq 2 && \forall\, \emptyset \subsetneq S \subsetneq V \\
& x_{uv} \geq 0 && \forall\, u, v \in V
\end{aligned}
$$

Subtour Elimination LP

[Dantzig, Fulkerson, Johnson '54]

$$\begin{aligned}
\min\ & \hat{c}^\top x \\
\text{s.t.}\ & \sum_{v} x_{uv} = 2 && \forall\, u \in V \\
& \sum_{u \in S, v \notin S} x_{uv} \geq 2 && \forall\, \emptyset \subsetneq S \subsetneq V \\
& x_{uv} \geq 0 && \forall\, u, v \in V
\end{aligned}$$

- Used in many approximation/exact algorithms for TSP.

Subtour Elimination LP

[Dantzig, Fulkerson, Johnson '54]

$$
\begin{aligned}
\min\ & \hat{c}^\top x \\
\text{s. t.}\ & \sum_{v} x_{uv} = 2 && \forall\, u \in V \\
& \sum_{u \in S, v \notin S} x_{uv} \geq 2 && \forall\, \emptyset \subsetneq S \subsetneq V \\
& x_{uv} \geq 0 && \forall\, u, v \in V
\end{aligned}
$$

- Used in many approximation/exact algorithms for TSP.
- The LP optimal value coincides with the Held–Karp bound.

Subtour Elimination LP

[Dantzig, Fulkerson, Johnson '54]

$$
\begin{aligned}
\min \ & \hat{c}^\top x \\
\text{s.t.} \ & \sum_{v} x_{uv} = 2 && \forall\, u \in V \\
& \sum_{u \in S, v \notin S} x_{uv} \geq 2 && \forall\, \emptyset \subsetneq S \subsetneq V \\
& x_{uv} \geq 0 && \forall\, u, v \in V
\end{aligned}
$$

- Used in many approximation/exact algorithms for TSP.
- The LP optimal value coincides with the Held–Karp bound.

Conjecture: The LP integrality gap is at most $4/3$ [Goemans '95].

2-ECSM LP

- LP relaxation of the 2-edge-connected spanning multisubgraph problem:

$$\begin{aligned} \min \;& c^\top x \\ \text{s. t.} \;& \sum_{e \in \delta_G(S)} x_e \geq 2 \qquad \forall\, \emptyset \subsetneq S \subsetneq V \\ & x_e \geq 0 \qquad \forall\, e \in E. \end{aligned}$$

2-ECSM LP

- LP relaxation of the 2-edge-connected spanning multisubgraph problem:

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t.} \sum_{e \in \delta_G(S)} & x_e \geq 2 \qquad \forall\, \emptyset \subsetneq S \subsetneq V \\ & x_e \geq 0 \qquad \forall\, e \in E. \end{aligned}$$

Fact: Subtour LP optimal value $=$ 2-ECSM LP optimal value.

[Cunningham '90] [Goemans, Bertsimas '93]

2-ECSM LP

- LP relaxation of the 2-edge-connected spanning multisubgraph problem:

$$\begin{aligned} \min \;\; & c^\top x \\ \text{s.t.} \;\; & \sum_{e \in \delta_G(S)} x_e \geq 2 \qquad \forall\, \emptyset \subsetneq S \subsetneq V \\ & x_e \geq 0 \qquad \forall\, e \in E. \end{aligned}$$

Fact: Subtour LP optimal value = 2-ECSM LP optimal value.

[Cunningham '90] [Goemans, Bertsimas '93]

- Methods for solving the LP:
  - **Ellipsoid:** separation oracle is min cut

2-ECSM LP

- LP relaxation of the 2-edge-connected spanning multisubgraph problem:

$$\begin{aligned} \min \ & c^\top x \\ \text{s.t.} \ & \sum_{e \in \delta_G(S)} x_e \geq 2 \qquad \forall\, \emptyset \subsetneq S \subsetneq V \\ & x_e \geq 0 \qquad \forall\, e \in E. \end{aligned}$$

Fact: Subtour LP optimal value = 2-ECSM LP optimal value.

[Cunningham '90] [Goemans, Bertsimas '93]

- Methods for solving the LP:
  - **Ellipsoid:** separation oracle is min cut
  - **Held–Karp bound/heuristic:** iterate over 1-trees

2-ECSM LP

- LP relaxation of the 2-edge-connected spanning multisubgraph problem:

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t.}\ & \sum_{e \in \delta_G(S)} x_e \geq 2 && \forall\, \emptyset \subsetneq S \subsetneq V \\ & x_e \geq 0 && \forall\, e \in E. \end{aligned}$$

Fact: Subtour LP optimal value = 2-ECSM LP optimal value.

[Cunningham '90] [Goemans, Bertsimas '93]

- Methods for solving the LP:
  - **Ellipsoid:** separation oracle is min cut
  - **Held–Karp bound/heuristic:** iterate over 1-trees
  - **Multiplicative weight update (MWU)**

Solving the LP via MWU

- FPTAS which returns a $(1+\varepsilon)$-approximate solution.

Solving the LP via MWU

- FPTAS which returns a $(1+\varepsilon)$-approximate solution.
- Sequential algorithms:
  - $\tilde{O}(n^4/\varepsilon^2)$ [Plotkin, Shmoys, Tardos '95]
  - $\tilde{O}(m^2/\varepsilon^2)$ [Garg, Khandekar '02]
  - $\tilde{O}(m/\varepsilon^2)$ [Chekuri, Quanrud '17]

Solving the LP via MWU

- FPTAS which returns a $(1+\varepsilon)$-approximate solution.
- Sequential algorithms:
  - $\tilde{O}(n^4/\varepsilon^2)$ [Plotkin, Shmoys, Tardos '95]
  - $\tilde{O}(m^2/\varepsilon^2)$ [Garg, Khandekar '02]
  - $\tilde{O}(m/\varepsilon^2)$ [Chekuri, Quanrud '17]

Main Result [KWY '25]

Parallel algorithm that runs in $\tilde{O}(m/\varepsilon^4)$ work and $\tilde{O}(1/\varepsilon^4)$ depth.

Solving the LP via MWU

- FPTAS which returns a $(1+\varepsilon)$-approximate solution.
- Sequential algorithms:
  - $\tilde{O}(n^4/\varepsilon^2)$ [Plotkin, Shmoys, Tardos '95]
  - $\tilde{O}(m^2/\varepsilon^2)$ [Garg, Khandekar '02]
  - $\tilde{O}(m/\varepsilon^2)$ [Chekuri, Quanrud '17]

Main Result [KWY '25]

Parallel algorithm that runs in $\tilde{O}(m/\varepsilon^4)$ work and $\tilde{O}(1/\varepsilon^4)$ depth.

Framework: Width-independent epoch-based MWU.

[Garg, Könemann '07] [Fleischer '00] [Luby, Nisan '93] [Young '01]

Epoch-Based MWU

- Initialize edge weights as $w = 1/c$.

Epoch-Based MWU

- Initialize edge weights as $w = 1/c$.
- Given a fixed lower bound λ on the mincut value, define

$$\mathcal{C}^* := \{C \text{ cut } : w(C) < (1+\varepsilon)\lambda\}.$$

Epoch-Based MWU

- Initialize edge weights as $w = 1/c$.
- Given a fixed lower bound λ on the mincut value, define

$$\mathcal{C}^* := \{C \text{ cut } : w(C) < (1+\varepsilon)\lambda\}.$$

While $\mathcal{C}^* \neq \emptyset$:

1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase w along these cuts.

an epoch

Epoch-Based MWU

- Initialize edge weights as $w = 1/c$.
- Given a fixed lower bound λ on the mincut value, define

$$\mathcal{C}^* := \{C \text{ cut } : w(C) < (1+\varepsilon)\lambda\}.$$

$$\left.\begin{array}{l}\textbf{While } \mathcal{C}^* \neq \emptyset\text{:}\\ \text{1. Select cut(s) from } \mathcal{C}^*.\\ \text{2. Multiplicatively increase } w \text{ along these cuts.}\end{array}\right\} \text{an epoch}$$

- $\lambda \leftarrow \lambda(1+\varepsilon)$ and a new epoch begins.

Epoch-Based MWU

- Initialize edge weights as $w = 1/c$.
- Given a fixed lower bound λ on the mincut value, define

$$\mathcal{C}^* := \{C \text{ cut } : w(C) < (1+\varepsilon)\lambda\}.$$

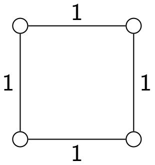
$$\left.\begin{array}{l}\textbf{While } \mathcal{C}^* \neq \emptyset\text{:}\\ \text{1. Select cut(s) from } \mathcal{C}^*.\\ \text{2. Multiplicatively increase } w \text{ along these cuts.}\end{array}\right\} \text{an epoch}$$

- $\lambda \leftarrow \lambda(1+\varepsilon)$ and a new epoch begins.
- Terminate when $\|w\|_\infty$ is big.

Epoch-Based MWU

While $\mathcal{C}^* \neq \emptyset$:

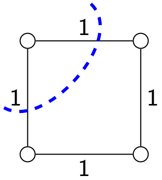
1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase $w^{(t)}$ along these cuts.



Epoch-Based MWU

While $\mathcal{C}^* \neq \emptyset$:

1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase $w^{(t)}$ along these cuts.

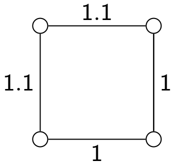


Sequential MWU: Select one cut from $\mathcal{C}^*$

Epoch-Based MWU

While $\mathcal{C}^* \neq \emptyset$:

1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase $w^{(t)}$ along these cuts.

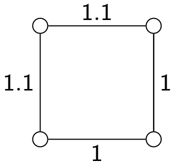


Sequential MWU: Select one cut from $\mathcal{C}^*$

Epoch-Based MWU

While $\mathcal{C}^* \neq \emptyset$:

1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase $w^{(t)}$ along these cuts.



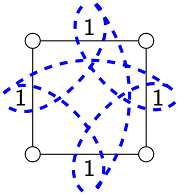
Sequential MWU: Select one cut from $\mathcal{C}^*$

$\implies \tilde{O}(m/\varepsilon^2)$ iterations [Garg, Könemann '07] [Fleischer '00].

Epoch-Based MWU

While $\mathcal{C}^* \neq \emptyset$:

1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase $w^{(t)}$ along these cuts.



Sequential MWU: Select one cut from $\mathcal{C}^*$

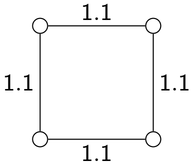
$\implies \tilde{O}(m/\varepsilon^2)$ iterations [Garg, Könemann '07] [Fleischer '00].

Parallel MWU: Select all cuts from $\mathcal{C}^*$

Epoch-Based MWU

While $\mathcal{C}^* \neq \emptyset$:

1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase $w^{(t)}$ along these cuts.



Sequential MWU: Select one cut from $\mathcal{C}^*$

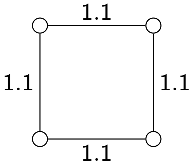
$\implies \tilde{O}(m/\varepsilon^2)$ iterations [Garg, Könemann '07] [Fleischer '00].

Parallel MWU: Select all cuts from $\mathcal{C}^*$

Epoch-Based MWU

While $\mathcal{C}^* \neq \emptyset$:

1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase $w^{(t)}$ along these cuts.



Sequential MWU: Select one cut from $\mathcal{C}^*$

$\implies \tilde{O}(m/\varepsilon^2)$ iterations [Garg, Könemann '07] [Fleischer '00].

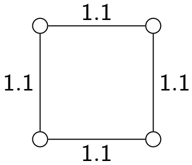
Parallel MWU: Select all cuts from $\mathcal{C}^*$

$\implies \tilde{O}(\log(|\mathcal{C}^*|)/\varepsilon^4)$ iterations [Luby, Nisan '93] [Young '01].

Epoch-Based MWU

While $\mathcal{C}^* \neq \emptyset$:

1. Select cut(s) from $\mathcal{C}^*$.
2. Multiplicatively increase $w^{(t)}$ along these cuts.



Sequential MWU: Select one cut from $\mathcal{C}^*$

$\implies \tilde{O}(m/\varepsilon^2)$ iterations [Garg, Könemann '07] [Fleischer '00].

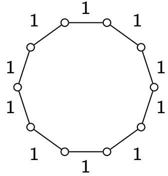
Parallel MWU: Select all cuts from $\mathcal{C}^*$

$\implies \tilde{O}(\log(|\mathcal{C}^*|)/\varepsilon^4)$ iterations [Luby, Nisan '93] [Young '01].

$\implies \tilde{O}(1/\varepsilon^4)$ iterations because $|\mathcal{C}^*| = O(n^2)$ for cuts.

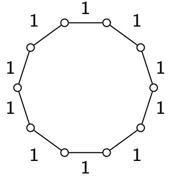
Core-Sequence

- Parallel MWU can incur $\Omega(n^2)$ work.

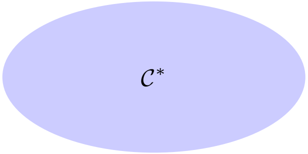


Core-Sequence

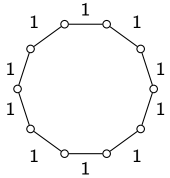
- Parallel MWU can incur $\Omega(n^2)$ work.



New Selection Rule:



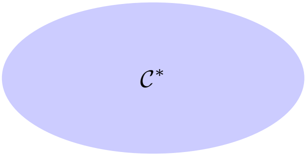
Core-Sequence



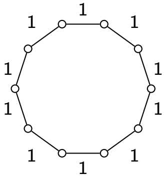
- Parallel MWU can incur $\Omega(n^2)$ work.

New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.



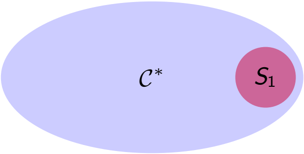
Core-Sequence



- Parallel MWU can incur $\Omega(n^2)$ work.

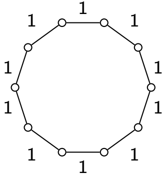
New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.



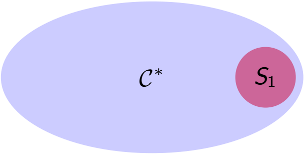
Core-Sequence

- Parallel MWU can incur $\Omega(n^2)$ work.



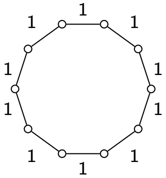
New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.



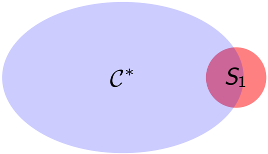
Core-Sequence

- Parallel MWU can incur $\Omega(n^2)$ work.

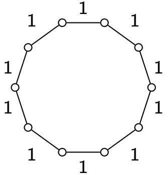


New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.



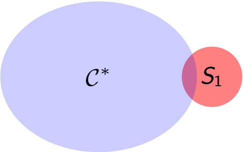
Core-Sequence



- Parallel MWU can incur $\Omega(n^2)$ work.

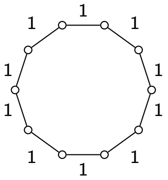
New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.



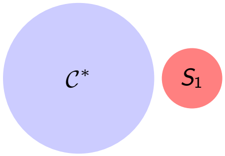
Core-Sequence

- Parallel MWU can incur $\Omega(n^2)$ work.



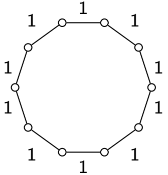
New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.



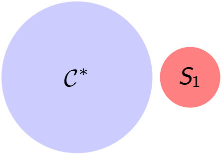
Core-Sequence

- Parallel MWU can incur $\Omega(n^2)$ work.



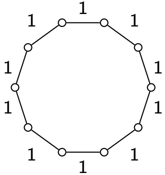
New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.
3. Repeat Steps 1 and 2 until $\mathcal{C}^* = \emptyset$.



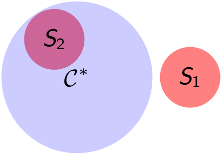
Core-Sequence

- Parallel MWU can incur $\Omega(n^2)$ work.

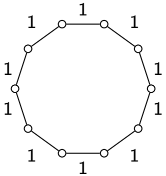


New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.
3. Repeat Steps 1 and 2 until $\mathcal{C}^* = \emptyset$.



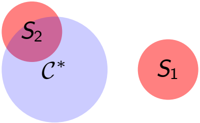
Core-Sequence



- Parallel MWU can incur $\Omega(n^2)$ work.

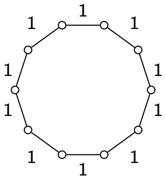
New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.
3. Repeat Steps 1 and 2 until $\mathcal{C}^* = \emptyset$.



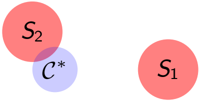
Core-Sequence

- Parallel MWU can incur $\Omega(n^2)$ work.

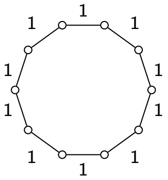


New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.
3. Repeat Steps 1 and 2 until $\mathcal{C}^* = \emptyset$.



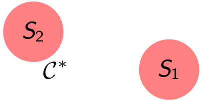
Core-Sequence



- Parallel MWU can incur $\Omega(n^2)$ work.

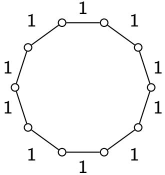
New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.
3. Repeat Steps 1 and 2 until $\mathcal{C}^* = \emptyset$.



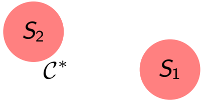
Core-Sequence

- Parallel MWU can incur $\Omega(n^2)$ work.



New Selection Rule:

1. Fix a representative set $S \subseteq \mathcal{C}^*$.
2. In every iteration, select $S \cap \mathcal{C}^*$ as long as it is nonempty.
3. Repeat Steps 1 and 2 until $\mathcal{C}^* = \emptyset$.



Definition

The sequence $\mathcal{S} = (S_1, \ldots, S_\ell)$ of representative sets is called a core-sequence of the epoch.

Core-Sequence

- Special cases:

Core-Sequence

- Special cases:
  - $\mathcal{S} = (S_1, \ldots, S_\ell)$ where $|S_i| = 1$ for all $i \in [\ell] \implies$ sequential MWU.

Core-Sequence

- Special cases:
  - $\mathcal{S} = (S_1, \ldots, S_\ell)$ where $|S_i| = 1$ for all $i \in [\ell] \implies$ sequential MWU.
  - $\mathcal{S} = (\mathcal{C}^*) \implies$ parallel MWU.

Core-Sequence

- Special cases:
  - $\mathcal{S} = (S_1, \ldots, S_\ell)$ where $|S_i| = 1$ for all $i \in [\ell] \implies$ sequential MWU.
  - $\mathcal{S} = (\mathcal{C}^*) \implies$ parallel MWU.

Theorem [KWY '25]

If MWU uses a core-sequence of length $\leq \ell$ with sets of size $\leq k$ in every epoch, then the number of iterations is

$$\tilde{O}\left(\frac{\ell \log(k)}{\varepsilon^4}\right).$$

Core-Sequence

- Special cases:
  - $\mathcal{S} = (S_1, \dots, S_\ell)$ where $|S_i| = 1$ for all $i \in [\ell] \implies$ sequential MWU.
  - $\mathcal{S} = (\mathcal{C}^*) \implies$ parallel MWU.

Theorem [KWY '25]

If MWU uses a core-sequence of length $\leq \ell$ with sets of size $\leq k$ in every epoch, then the number of iterations is

$$\tilde{O}\left(\frac{\ell \log(k)}{\varepsilon^4}\right).$$

- Tradeoff between ℓ and k.

Core-Sequence for 2-ECSM LP

Theorem [KWY '25]

For 2-ECSM LP, every epoch has a core-sequence of length $\tilde{O}(1)$, in which every set has size $\tilde{O}(n)$.

Core-Sequence for 2-ECSM LP

Theorem [KWY '25]

For 2-ECSM LP, every epoch has a core-sequence of length $\tilde{O}(1)$, in which every set has size $\tilde{O}(n)$.

- Despite $|\mathcal{C}^*| = O(n^2)$, only need to select $\tilde{O}(n)$ of them!

Core-Sequence for 2-ECSM LP

Theorem [KWY '25]

For 2-ECSM LP, every epoch has a core-sequence of length $\tilde{O}(1)$, in which every set has size $\tilde{O}(n)$.

- Despite $|\mathcal{C}^*| = O(n^2)$, only need to select $\tilde{O}(n)$ of them!

Theorem [KWY '25]

There is a parallel FPTAS for the Held–Karp bound that runs in $\tilde{O}(m/\varepsilon^4)$ work and $\tilde{O}(1/\varepsilon^4)$ depth.

Finding a Good Core-Sequence

- $\tilde{O}(1)$ sets
- Every set has size $\tilde{O}(n)$

Tree Packing

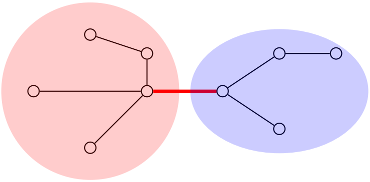
Definition

Fix a spanning tree T of G. A cut C k-respects T if $|C \cap E(T)| = k$.

Tree Packing

Definition

Fix a spanning tree T of G. A cut C k-respects T if $|C \cap E(T)| = k$.

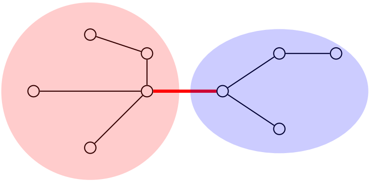


1-respecting cut

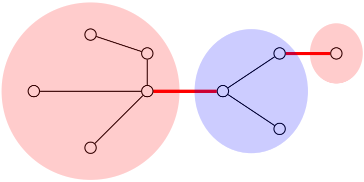
Tree Packing

Definition

Fix a spanning tree T of G. A cut C k-respects T if $|C \cap E(T)| = k$.



1-respecting cut

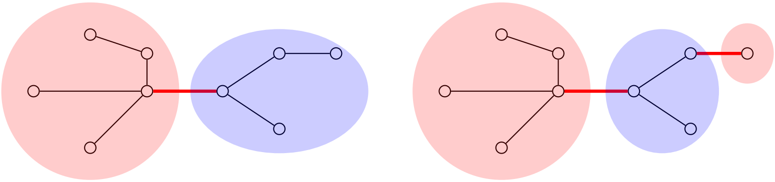


2-respecting cut

Tree Packing

Definition

Fix a spanning tree T of G. A cut C *k-respects* T if $|C \cap E(T)| = k$.



1-respecting cut

2-respecting cut

Theorem [Karger '00]

There exists a family $\mathcal{T}$ of $O(\log n)$ spanning trees such that w.h.p., every cut in $\mathcal{C}^*$ 1- or 2-respects some $T \in \mathcal{T}$.

Reducing to a Tree

- For every tree $T \in \mathcal{T}$, let

$$\mathcal{C}^*_T := \{C \in \mathcal{C}^* : |C \cap E(T)| \leq 2\}.$$

Reducing to a Tree

- For every tree $T \in \mathcal{T}$, let

$$\mathcal{C}^*_T := \{C \in \mathcal{C}^* : |C \cap E(T)| \leq 2\}.$$

- By Karger's Theorem, it suffices to find a good core-sequence for $\mathcal{C}^*_T$.

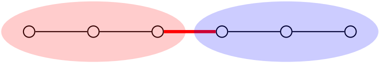
Reducing to a Tree

- For every tree $T \in \mathcal{T}$, let

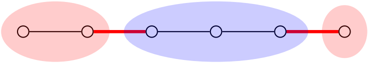
$$\mathcal{C}^*_T := \{C \in \mathcal{C}^* : |C \cap E(T)| \leq 2\}.$$

- By Karger's Theorem, it suffices to find a good core-sequence for $\mathcal{C}^*_T$.

- For simplicity, assume that T is a path.



1-respecting cut



2-respecting cut

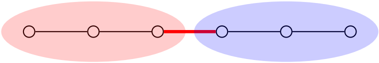
Reducing to a Tree

- For every tree $T \in \mathcal{T}$, let

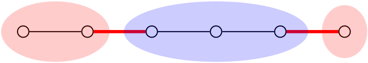
$$\mathcal{C}_T^* := \{C \in \mathcal{C}^* : |C \cap E(T)| \leq 2\}.$$

- By Karger's Theorem, it suffices to find a good core-sequence for $\mathcal{C}_T^*$.

- For simplicity, assume that T is a path.



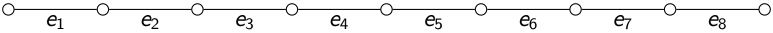
1-respecting cut



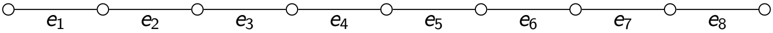
2-respecting cut

- $\mathcal{C}_T^*$ can still be as large as $O(n^2)$.

2-Respecting Cuts



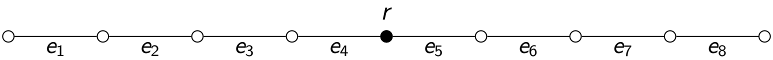
2-Respecting Cuts



Definition

Fix a root $r \in V$. A 2-respecting cut $\{e_i, e_j\}$ crosses r if r lies on the subpath between e_i and e_j.

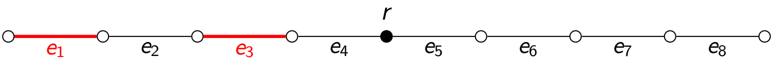
2-Respecting Cuts



Definition

Fix a root $r \in V$. A 2-respecting cut $\{e_i, e_j\}$ crosses r if r lies on the subpath between e_i and e_j.

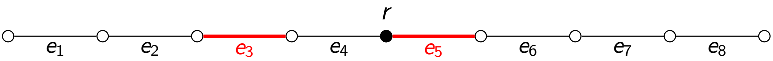
2-Respecting Cuts



Definition

Fix a root $r \in V$. A 2-respecting cut $\{e_i, e_j\}$ crosses r if r lies on the subpath between e_i and e_j.

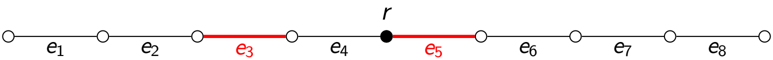
2-Respecting Cuts



Definition

Fix a root $r \in V$. A 2-respecting cut $\{e_i, e_j\}$ crosses r if r lies on the subpath between e_i and e_j.

2-Respecting Cuts



Definition

Fix a root $r \in V$. A 2-respecting cut $\{e_i, e_j\}$ crosses r if r lies on the subpath between e_i and e_j.

***r*-Crossing Lemma**

If every 2-respecting cut in $\mathcal{C}_T^*$ crosses r, then $|\mathcal{C}_T^*| = O(n)$.

Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges to the left and right of r respectively.

Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges
to the left and right of r respectively.

- Define a matrix $M \in \{0,1\}^{k \times \ell}$ as

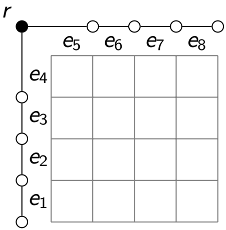
$$M_{e_i,e_j} := \begin{cases} 1, & \text{if } \{e_i, e_j\} \in \mathcal{C}_T^* \\ 0, & \text{otherwise.} \end{cases}$$

Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges to the left and right of r respectively.

- Define a matrix $M \in \{0, 1\}^{k \times \ell}$ as

$$M_{e_i, e_j} := \begin{cases} 1, & \text{if } \{e_i, e_j\} \in \mathcal{C}_T^* \\ 0, & \text{otherwise.} \end{cases}$$

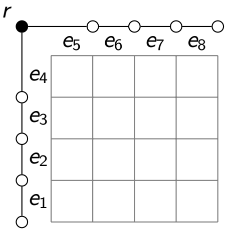


Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges to the left and right of r respectively.

- Define a matrix $M \in \{0,1\}^{k\times\ell}$ as

$$M_{e_i,e_j} := \begin{cases} 1, & \text{if } \{e_i, e_j\} \in \mathcal{C}_T^* \\ 0, & \text{otherwise.} \end{cases}$$



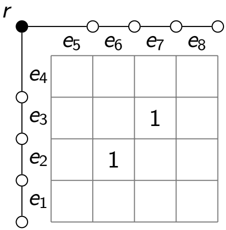
Claim: Every anti-diagonal of M has at most one 1.

Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges to the left and right of r respectively.

- Define a matrix $M \in \{0,1\}^{k \times \ell}$ as

$$M_{e_i, e_j} := \begin{cases} 1, & \text{if } \{e_i, e_j\} \in \mathcal{C}_T^* \\ 0, & \text{otherwise.} \end{cases}$$



Claim: Every anti-diagonal of M has at most one 1.

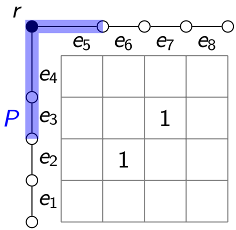
- Suppose there are two 1's in some anti-diagonal.

Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges to the left and right of r respectively.

- Define a matrix $M \in \{0,1\}^{k\times\ell}$ as

$$M_{e_i,e_j} := \begin{cases} 1, & \text{if } \{e_i, e_j\} \in \mathcal{C}_T^* \\ 0, & \text{otherwise.} \end{cases}$$



Claim: Every anti-diagonal of M has at most one 1.

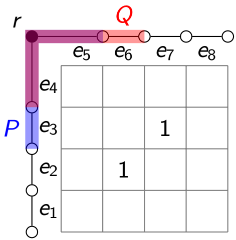
- Suppose there are two 1's in some anti-diagonal.

Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges to the left and right of r respectively.

- Define a matrix $M \in \{0,1\}^{k \times \ell}$ as

$$M_{e_i,e_j} := \begin{cases} 1, & \text{if } \{e_i, e_j\} \in \mathcal{C}_T^* \\ 0, & \text{otherwise.} \end{cases}$$



Claim: Every anti-diagonal of M has at most one 1.

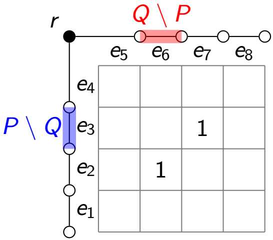
- Suppose there are two 1's in some anti-diagonal.

Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges to the left and right of r respectively.

- Define a matrix $M \in \{0,1\}^{k \times \ell}$ as

$$M_{e_i,e_j} := \begin{cases} 1, & \text{if } \{e_i, e_j\} \in \mathcal{C}_T^* \\ 0, & \text{otherwise.} \end{cases}$$



Claim: Every anti-diagonal of M has at most one 1.

- Suppose there are two 1's in some anti-diagonal.

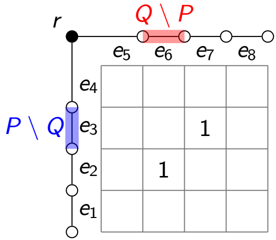
$$2(1+\varepsilon)\lambda > w(\delta_G(P)) + w(\delta_G(Q)) \geq w(\delta_G(P \setminus Q)) + w(\delta_G(Q \setminus P))$$

Proof of r-Crossing Lemma

- Let k and ℓ be the number of edges to the left and right of r respectively.

- Define a matrix $M \in \{0,1\}^{k \times \ell}$ as

$$M_{e_i, e_j} := \begin{cases} 1, & \text{if } \{e_i, e_j\} \in \mathcal{C}_T^* \\ 0, & \text{otherwise.} \end{cases}$$



Claim: Every anti-diagonal of M has at most one 1.

- Suppose there are two 1's in some anti-diagonal.

$$2(1+\varepsilon)\lambda > w(\delta_G(P)) + w(\delta_G(Q)) \geq w(\delta_G(P \setminus Q)) + w(\delta_G(Q \setminus P))$$

$\implies$ $P \setminus Q$ or $Q \setminus P$ belongs to $\mathcal{C}_T^*$. $\square$

Good Core-Sequence for $\mathcal{C}_T^*$

Good Core-Sequence for $\mathcal{C}_T^*$

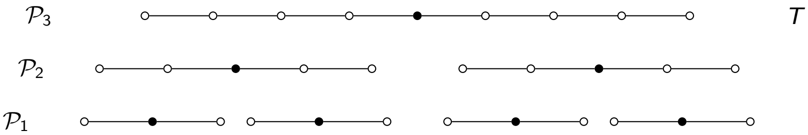
- First set in the core-sequence

$$S_0 := \{C \in \mathcal{C}_T^* : |C \cap E(T)| = 1\}.$$

Good Core-Sequence for $\mathcal{C}_T^*$

- First set in the core-sequence

$$S_0 := \{C \in \mathcal{C}_T^* : |C \cap E(T)| = 1\}.$$

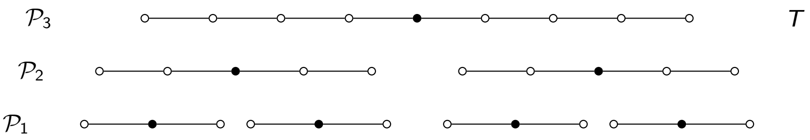


- For $i \geq 1$, decompose T into paths $\mathcal{P}_i$ of length 2^i.

Good Core-Sequence for $\mathcal{C}_T^*$

- First set in the core-sequence

$$S_0 := \{C \in \mathcal{C}_T^* : |C \cap E(T)| = 1\}.$$



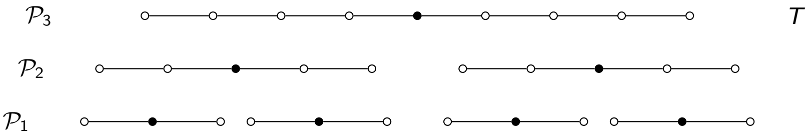
- For $i \geq 1$, decompose T into paths $\mathcal{P}_i$ of length 2^i. Set

$$S_i := \{C \in \mathcal{C}_T^* : |C \cap E(P)| = 2 \text{ for some } P \in \mathcal{P}_i\}.$$

Good Core-Sequence for $\mathcal{C}_T^*$

- First set in the core-sequence

$$S_0 := \{C \in \mathcal{C}_T^* : |C \cap E(T)| = 1\}.$$



- For $i \geq 1$, decompose T into paths $\mathcal{P}_i$ of length 2^i. Set

$$S_i := \{C \in \mathcal{C}_T^* : |C \cap E(P)| = 2 \text{ for some } P \in \mathcal{P}_i\}.$$

- By r-crossing lemma, $|S_i| = O(n)$ for all i.

Conclusion

- Introduced core-sequence as a new selection rule for MWU.

Conclusion

- Introduced core-sequence as a new selection rule for MWU.
- Parallel FPTAS that runs in nearly linear work and polylog depth for
  - Held–Karp bound and k-ECSM LP
  - k-ECSS LP

Conclusion

- Introduced core-sequence as a new selection rule for MWU.
- Parallel FPTAS that runs in nearly linear work and polylog depth for
  - Held–Karp bound and k-ECSM LP
  - k-ECSS LP
- Future directions:
  - Apply core-sequence to other implicit packing/covering LPs

Conclusion

- Introduced core-sequence as a new selection rule for MWU.
- Parallel FPTAS that runs in nearly linear work and polylog depth for
  - Held–Karp bound and k-ECSM LP
  - k-ECSS LP
- Future directions:
  - Apply core-sequence to other implicit packing/covering LPs
  - Better dependence on ε

Conclusion

- Introduced core-sequence as a new selection rule for MWU.
- Parallel FPTAS that runs in nearly linear work and polylog depth for
  - Held–Karp bound and k-ECSM LP
  - k-ECSS LP
- Future directions:
  - Apply core-sequence to other implicit packing/covering LPs
  - Better dependence on ε
  - Extension to streaming/distributed models

Conclusion

- Introduced core-sequence as a new selection rule for MWU.
- Parallel FPTAS that runs in nearly linear work and polylog depth for
  - Held–Karp bound and k-ECSM LP
  - k-ECSS LP
- Future directions:
  - Apply core-sequence to other implicit packing/covering LPs
  - Better dependence on ε
  - Extension to streaming/distributed models

Thank You!