Accelerating the MwU Framework for Packing LPs

Zhuan (Cedric) Khye-Koh
CWI Amsterdam

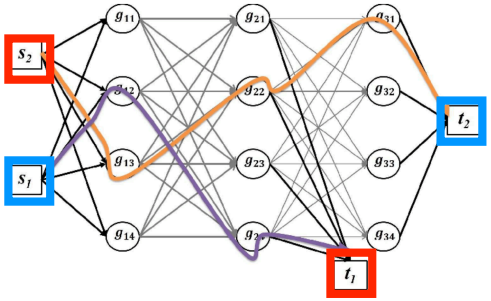
Omri Weinstein
Hebrew U & Columbia U

Sorrachai Yingchareonthawornchai
Simons → Hebrew U

European Research Council

Network Design via Packing LPs

- Many network design & combinatorial opt. problems formulated fractional Packing LPs:
- Ex: Multicommodity Flows, Matching, k-Connectivity, Steiner trees, Linear Arrangement…



$$\max \sum_{p \in \mathcal{P}} f_p$$
$$\text{s.t.} \sum_{p: p \ni e} f_p \le c(e) \quad \forall e \in E,$$
$$f_p \ge 0 \quad \forall p \in \mathcal{P}.$$

$$\min \sum_{e \in E} w(e)c(e)$$
$$\text{s.t.} \sum_{e \in p} w(e) \ge 1 \quad \forall p \in \mathcal{P}$$
$$w(e) \ge 0 \qquad \forall e \in E$$

Dual = Fractional cut:
assigning weights w() ≥ 0 to edges
length ($\forall$ (s_i, t_i)-Path $\in \mathcal{P}$) ≥ 1 .
Certifies >1 unit s-t-flow req > 1 vol.

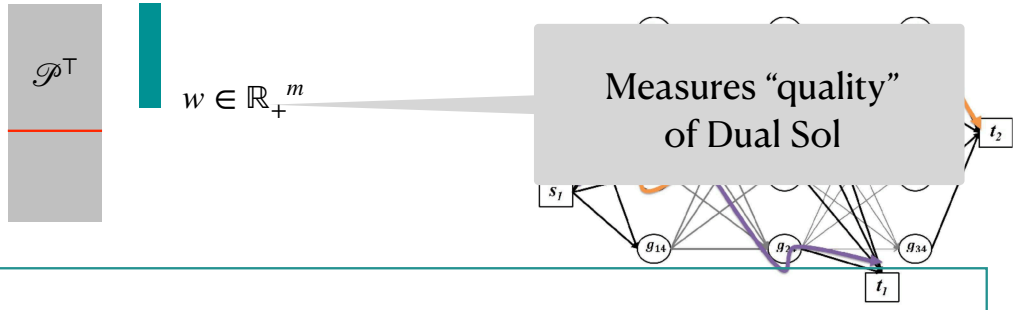
- k-CF : $\mathcal{P}$ = set of (s_i, t_i)-paths for k demand-pairs
- (primal = exp(n) vars, but *dual* = m vars!)

- [GK'98, F'00] $(1+\epsilon)$ *width-independent algorithm*: $O(m^2/\epsilon^2)$ + output-size ($\Omega(kn)$) time, via primal-dual MwU [precursors: KPST94, Young'05,Chekuri et al.,..].

- *High-accuracy* k-CF ≡ General LPs ($\Omega(mn + n^\omega)$) (even for k=2! [KZ20]).
  - For small k: O(mk/ϵ). [Sherman'17]

PD-MwU Framework for apx Packing LPs [Garg-Konemann'98]

- Assume unit capacities $c(e) \equiv 1$.
- **Oracle**: $\text{MinRow}_{\mathcal{P}}(w_t) := \arg\min_{p \in \mathcal{P}} \langle p, w_t \rangle$



- Input: G=(n,m) , k demand pairs $\{(s_i, t_i)\}_{i \in [k]}$
- Initialize $f \leftarrow \emptyset$, increasing weight function $w_t()$, init $w_0(e) \leftarrow 1 \quad \forall e \in \mathbb{R}^E$.
- While no edge $e \in E$ with $\text{cong}_t(e) > \log m/\epsilon^2$, do :

$$cong_t(e) := \sum_{p:\, e \in p} \frac{f_p^{(t)}}{c(e)} = \sum_{p:\, e \in p} f_p^{(t)}$$

  - p* $\leftarrow$ Min-Row$(w_t) := \arg\min_{p \in \mathcal{P}} \langle p, w_t \rangle$ // find shortest-path w.r.t $w_t()$
  - Augment $f^{(t)}$ with 1 unit of flow through p*. (in gen, c_{min} = bottleneck capacity)
  - Increase weight $\forall\, e \in p^* : w_{t+1}(e) \leftarrow w_t(e) \cdot \exp(\epsilon)$ (in gen, $w(e) \cdot \exp(\frac{\epsilon \cdot c_{min}}{c(e)})$)
- Return $f^{(T)}$.

$$w_t(e) \propto \exp(\epsilon \cdot \text{cong}_t(e))$$

6-Line Analysis (based on [Y'18])

Termination cond: $\text{cong}_t(e) > \log m/\epsilon^2$

- Flow produced by MwU may violate capacities ⇒ Scale $f^{(T)}$ by max-congestion of $e \in E$.
- *Obs:* max-cong $\approx \log m/\epsilon^2$ (UB since $w_t(e) \propto \exp(\epsilon \cdot \text{cong}_t(e))$ ("softmax" argument)
- Theorem: Scaled-down flow $f^{(T)}$ computed by MwU satisfies: $\frac{|f^{(T)}|}{\text{max-cong}} \geq (1-\varepsilon)OPT$.

Sketch: Def $\Phi_t := \sum_e w_t(e)$, $D_t := \Phi_t / w_t(p_t^*)$ (feasible) ⇒ **Claim**: $\Phi_{t+1} \approx \Phi_t \cdot \exp(\epsilon/D_t)$.

$$\Rightarrow \Phi_T \leq m \cdot \exp\left(\sum_t^T \frac{\epsilon}{D_t}\right) \leq m \exp\left(\frac{\epsilon}{D_{\min}} \sum_{j=1}^{T} \min_{\sum_{e\in E} w(e)c(e)} \text{s.t.} \sum_{e\in p} w(e) \geq 1 \quad \forall p \in \mathcal{P}\right) = m \exp\left(\epsilon \cdot \frac{f^{(T)}}{D_{\min}}\right)$$

(since ∀ round route 1 unit

On other hand, $\Phi_T \geq \max_e w_T(e) = \exp(\epsilon \cdot \text{max-cong}_T) \leq \epsilon \cdot \frac{f^{(T)}}{D_{\min}} + \log m$

$$\Leftrightarrow 1 \leq \frac{f^{(T)}/\text{maxcong}_T}{D_{\min}} + \frac{\log m}{\text{maxcong}_T} \Leftrightarrow \frac{f^{(T)}/\text{maxcong}_T}{D_{\min}} \geq 1 - \frac{\log m}{\epsilon \cdot \text{maxcong}_T} \geq 1 - \epsilon$$

- **# Iterations:** $T \leq \sum_e \text{cong}_e^{(T)} \leq O(m \cdot \log m/\epsilon^2)$ (Tight)

($\forall$ iter routs ≥ 1 unit flow through some edge, no edge can exceed $O(\log m/\epsilon^2)$)

- Cost(MinRow(w_t)) = Cost(k-APSP): $O(m \min\{k, n\})$ (SSSP from all n vertices via Dijkstra)

- [Fleischer00]. $O(m)$ amortized time $\Rightarrow$ $O(m^2/\epsilon^2)$ total runtime k-CF.

- *Obs 1:* Nothing changes in analysis if every iter computes $(1+\epsilon)$-MinRow $= (1+\epsilon)$-APSP

- Obs 2 ("Epoch-Based" MwU) [Fleischer'00]: Can group MwU iterations into epochs $E_i := [\lambda, (1+\epsilon)\lambda]$. **Goal**($E_i$) : Clear all demands in $E_i \Leftrightarrow$ Compute **all paths** of weight $w_t \in E_i$, update repeatedly until *all residual* SPs in G have length $> (1+\epsilon)\lambda$.

- Claim [Fleischer00]: Epoch-MwU Algo terminates after $O(\log m / \epsilon^2)$ *Epochs*.

- Key implication of Epoch-Based MwU $\Rightarrow$ Cycling through n demands **at most once** per Epoch $\Rightarrow$ **Total** O(n) calls to Dijkstra (SSSP) = O(m(k+n)) **total** time per Epoch.

Speedup via Decremental DS [Madry'10]

- [Madry'10]: $O(m^2/\epsilon^2) \rightarrow O(mn/\epsilon^2)$ time by speeding-up *cost-per-iteration*.
- Main obs: MinRow() Oracle calls = decremental $(1+\epsilon)$-APSP, classic DS (EvenShiloach-Trees : work well for decremental SSSP with low diameter).
- "Black-box" Dec APSP DS doesn't suffice to get O(n) time per iteration (ES trees have O(m) query for SP *reporting*) — **Sparsification** (SP "hitting-set") ⇒ O(n) amortized per iter.
- $O(mn)$ is a natural barrier ("flow-decomposition barrier" [M
- $O(mn)$ also state-of-art for many network-design problems [Fleishcer et al.]

Major driving motivation for decremental APSP (Aaron's talk)

Reducing MwU Iterations via Min-Row-**Packing** [KWY'23]

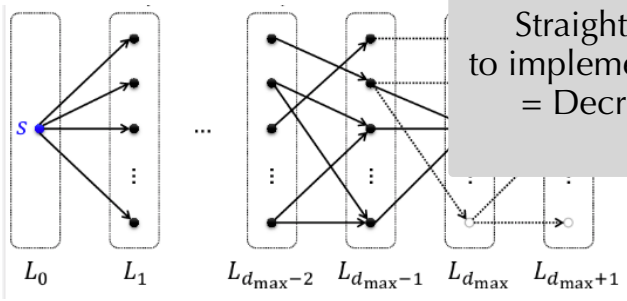
- Recall: MwU Algo terminates after $O(\log m / \epsilon^2)$ Epochs, *where* Goal(E_i) = Compute **all SPs** of weight $w_t \in [\lambda, (1+\epsilon)\lambda]$, and update repeatedly until *all residual* SPs in G $> (1+\epsilon)\lambda$.

- Maximal **SP-Packing** (MSP Oracle): Given w_t-weighted G, return maximal edge-disjoint paths $\mathscr{M}_t$ of weight $w_t \in [\lambda, (1+\epsilon)\lambda]$.
  (Maximal ⇔ Every residual SP in range must intersect $\mathscr{M}_t$).

Thm [KWY'23]: Epoch-MwU Alg with MSP Oracle terminates in $\Theta_\epsilon(\sqrt{m}\log m)$ iterations.

- Prototype example: Packing **(s,t)-Min-Cuts**

(similar idea to Blocking Flows: Run DFS on residual graph G_f until get "stuck", contract all edges in conn comp. & proceed.)

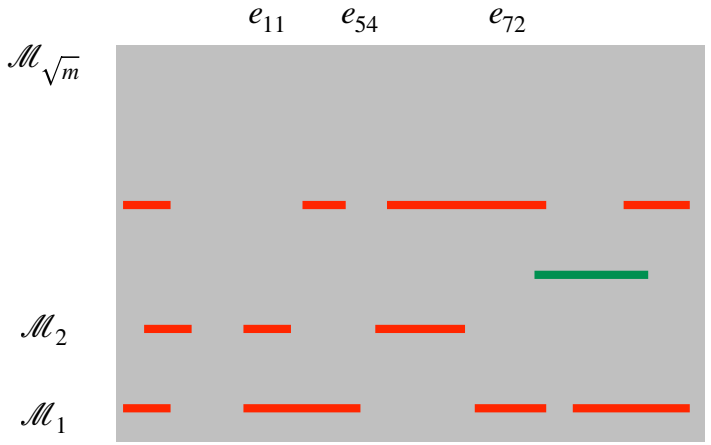


Straightforward way to implement MSP Orcale = Decremental SPs

Proof of MSP Theorem

Thm [KWY'23]: Epoch MwU with MRP Oracle terminates in $\Theta(\sqrt{m}\log(m)/\epsilon^2)$ iteration.

- **Correctness**: Immediate from Epoch-based analysis: MwU updates along SP, but paths in $\mathscr{M}_t$ disjoint $\Rightarrow$ augmenting all flows in $\mathscr{M}_t$ in parallel is equivalent to seq Epoch.
- **Iteration conplexity**: (unit-capacity case) $\Rightarrow$ **terminate after $O(\sqrt{m}/\epsilon^2)$ rounds**
- Case I : $\forall\, t \in [\sqrt{m}]$, $|\text{Supp}(\mathscr{M}_t)| \geq \Omega(\sqrt{m})$.
- Case 2 : $\exists\, t \in [\sqrt{m}]$ $s.t$ $|\text{Supp}(\mathscr{M}_t)| < \sqrt{m}$.
  - Use MSP property: $\forall\, t_j > t_i$, $\mathscr{M}_i \cap \mathscr{M}_j \neq \emptyset$:
- $O(1/\epsilon^4) \to O(1/\epsilon^3)$**:** Refine Supp $\epsilon\sqrt{m}$, local/global arg for small/large support case.



- General capacities case more involved (MSP def more intricate, ($O(\log C_{\max})$ iter overhead).

Accelerating Network Design LPs via the MSP Oracle

- MSP Oracle gives *tradeoff* b/w # iterations & cost-per-iteration.
- **Upshot**: Speedup over MwU whenever Packing MinRow $\ll$ Decremental MinRow.
- For k-CF $\Rightarrow$ $(1+\epsilon)$-*SP Packing* $\ll (1+\epsilon)$-*Decremental APSP* [Madry'10] ??
- Unfortunately, No ($(1+\epsilon)$-Closest Pair) $\Omega(mn)$ under OV (reduction from *shortest cycle* problem [Williams et al. '21]) ($\Rightarrow$ Real barrier for o(mn) MCF via MwU framework..)
- C*an* get $m^{1+o(1)}$ time ($\ll mn$) MCF for clique demands [KWY'23] (using isolating-cuts)
- **k-Edge Connectivity** (k-ECC):
  Input: (weighted) G, terminal set T$\subset$V, Goal: find (induced) subgraph $H \subset G$
  s.t *every Steiner cut* (separating T) has size size $\geq k$. ($\Leftrightarrow$ k-edge-disjoint paths)

best known: O(mn) via GK-MwU framework + Ad-hoc Decremental DS [CHNSSY '22]

Packing Maximal Steiner Cuts (towards $\tilde{O}(m^{1.5})$-time algo for k-ECC)

- Packing LP for k-ECC $\mathscr{C}$ = Steiner-Cuts(T) $\Rightarrow$ $\text{MinRow}_{\mathscr{C}}(w_t) := \min_{C \in \mathscr{C}} \langle c, w_t \rangle \iff$

Min Steiner-Cut (MSC): Find a (ϵ-apx) minimum-weight Steiner cut separating T w.r.t $w_t()$.

Min Steiner-Cut **Packing** (MSCP): Find a **maximal** disjoint collection of Steiner cuts $\subset G$ of weight in given range $w_t(C) \in [\lambda, (1+\epsilon)\lambda]$.

- Promising Prerequisite **(*)**: Packing **(s,t)**-MSCP can be done in MaxFlow time $(m^{1+o(1)})$.

- [KWY'23] **Exact** MSCP can be done in MaxFlow time $(m^{1+o(1)})$
  (via packing Isolating Cuts a-la [AKLPST'23] and then invoking (*) on log|T| (s-t)-pairs…

- Packing **Apx** MSCP is a different ballgame.. ((*) may miss some cuts in ran
  - Argument goes through Fair-Cuts [LNPS'23] (ongoing work..)

Jason: Packing Steiner TREES in $m^{1+o(1)}$

Conclusions & Open Questions

- Find more graph problems where Cost(MRP) $\ll$ Decremental MinRow Problem ?
- Can MRP framework be extended to **any** positive LP (mixed packing/covering)? (Current analysisassumes constraint matrix is binary).
- Improve ϵ-dependence of # MSP Iterations from $1/\epsilon^3 \rightarrow 1/\epsilon^2$ (Not easy)

Thanks (esp. to all organizers)!