Approximating the Held-Karp bound for Metric-TSP in Near-linear Time

Chandra Chekuri Kent Quanrud

Univ. of Illinois, Urbana-Champaign

HALG 2018

TSP and Metric-TSP

- **TSP:** Undir graph G=(V,E), edge costs c_e
  - find *Hamiltonian Cycle* in G of minimum cost
  - Inapproximable
- **Metric-TSP:** Undir graph G=(V,E), edge costs c_e
  - find *spanning tour/closed walk* in G of minimum cost
  - same as Hamiltonian Cycle in *metric completion* of G
  - APX-Hard and constant factor approximation known

Metric-TSP

- **Explicit representation:** complete graph on n nodes with all $\Omega(n^2)$ distances explicitly specified
- **Implicit representation:** weighted graph on n nodes and m edges. Shortest path distances define the metric

This talk: implicit representation

Approximating Metric-TSP

- 3/2 approximation: Christofides heuristic (1976)
- **Conjecture:** 4/3 approximation via LP relaxation
- Recent exciting progress on graphic-TSP, s-t-Path-TSP, ATSP etc. Mostly based on **LP relaxations.**

Subtour Elimination LP for TSP

[Dantzig-Fulkerson-Johnson 1954]

$$\min \sum_{e \in E}^{n} c_e x_e$$

$$\begin{aligned} x(\delta(v)) &= 2 && v \in V \\ x(\delta(S)) &\geq 2 && \emptyset \subset S \subset V \\ x_e &\geq 0 && e \in E \end{aligned}$$

2ECSS LP

$$\min \sum_{e \in E} c_e x_e$$

$$\begin{aligned} x(\delta(S)) &\geq 2 & \emptyset \subset S \subset V \\ x_e &\geq 0 & e \in E \end{aligned}$$

Metric-TSP: 2ECSS LP is equivalent to Subtour LP [Cunningham, Bertsimas-Goemans]

Solving the LP

- **Ellipsoid:** separation oracle is global mincut
- **Held-Karp bound/algorithm:** iterative method that converges to the LP solution
- **FPTASes via MWU/Lagrangean relaxation**
  - $(1+\varepsilon)$ approximation
  - $O(m^2 \log^4 n/\varepsilon^2)$ randomized [Plotkin-Shmoys-Tardos'95] via Karger's mincut algorithm
  - $O(m^2 \log^2 n/\varepsilon^2)$ [Garg-Khandekar'02]

Solving the LP

- **FPTASes via MWU/Lagrangean relaxation**
  - $O(m^2 \log^4 n/\varepsilon^2)$ randomized [Plotkin-Shmoys-Tardos'95]
  - $O(m^2 \log n/\varepsilon^2)$ [Garg-Khandekar'02]

- **Theorem** [C-Quanrud FOCS 2017] Randomized algorithm that runs in $O(m \log^4 n/\varepsilon^2)$ time.

High-level MWU Framework for *Implicit* Problems

- Speed up classical MWU based approximation schemes for implicit problems
- Problem-specific **integration** of *dynamic data structures* for two separate issues
  - oracle for MWU
  - (lazy) weight update

Inspired by [Koufagiannis-Young'07, Young'15] for weight update, and other work on MWU [Madry'10, Agarwal-Pan'14 …]

Other Applications

- Packing spanning trees
- Geometric packing/covering
- k-cuts
- Covering integer programs via knapsack cover inequalities
- Mixed packing and covering
- …

2ECSS LP

$$\min \sum_{e \in E} c_e x_e$$

$$\begin{aligned} x(\delta(S)) &\geq 2 & \emptyset \subset S \subset V \\ x_e &\geq 0 & e \in E \end{aligned}$$

Dual of 2ECSS LP

$$\max 2 \sum_{S} y_S$$

$$\sum_{S:e \in \delta(S)} y_S \leq c_e \qquad e \in E$$

$$y_S \geq 0 \qquad S \subset V$$

Packing cuts into capacities

MWU Approach

- Maintain positive weights for *constraints*: w_e for edge e. Initialize $w_e = 1/c_e$
- Maintain current feasible solution x, initialized to **0**
- Each iteration:
  - **solve mincut** in G with edge weights w_e
  - take small step according to mincut: $x = x + \gamma\, 1_{\delta(S^*)}$
  - **update weights**: exponential in load. $w_e = (1/c_e)\exp(\eta\, x_e/c_e)$
- Iterate until done

MWU Approach

- Each iteration
  - requires solving a global mincut problem: randomized $O(m \log^3 n)$ alg [Karger'00]
  - Update weights of up to m edges
- $O(m \log m/\varepsilon^2)$ iterations via MWU analysis
- $O(m^2 \log^4 m/\varepsilon^2)$ randomized algorithm [Plotkin-Shmoys-Tardos'95]
- $O(m^2 \log^2 m/\varepsilon^2)$ deterministic algorithm based on arborescence packing [Garg-Khandekar'04]

Our Approach

- Each iteration requires solving a global mincut problem: randomized $O(m \log^3 n)$ alg [Karger'00]
- $O(m \log m/\varepsilon^2)$ iterations

Two ideas:

1. Make Karger's algorithm *incremental*
2. Weight update *integrated* with incremental mincut algorithm

Overall time is is about $O(\log n)$ mincut computations

(Standard) MWU Advantages

- $(1+\varepsilon)$ *approximation* for mincut suffices
- weights can be maintained to within $(1+\varepsilon)$ factor – can be updated *lazily*
- Weights *increase monotonically*
- MWU analysis: each edge's weight increases by multiplicative $(1+\varepsilon)$ factor only $O(\log m/\varepsilon^2)$ times

Weight update bottleneck

Suppose computing mincut was **free!**

Oracle *outputs* edges of mincut

Bottleneck: updating weights of all edges in mincut

Solution:

- Amortization/randomization to charge for update
- Need edges of mincut to be presented *implicitly*

Weight update amortization

Suppose mincut has 4 edges $e_{10}, e_{23}, e_{78}, e_{85}$ with capacities 5, 100, 3500, 7 respectively

Recall: $w_e = (1/c_e)\exp(\eta\, x_e/c_e)$, $w'_e = w_e \exp(\gamma\eta/c_e)$

Update to e_{23}, e_{78} significantly smaller in terms of multiplicative factor.

Update e_{10}, e_{85} and charge time to $(1+\varepsilon)$-factor increase

Update e_{23}, e_{78} probabilistically or deterministically via amortized data structure [Young'15,C-Quanrud'17,'18]

Randomized weight update

- Pick random $\theta \in [0,1]$
- For each edge e in (approximate) mincut
  - If $\theta \leq cm_{in}/c_e$ then $w_e = \exp(\epsilon)\ w_e$
  - Else w_e remains the same
- Updates are **"correlated"** and hence easy to implement
- **Catch:** need to prove that MWU analysis works with correlated rounding - drift analysis [KY'07,CQ'18]

Karger's near-linear time mincut algorithm

[STOC'96, JACM'00]

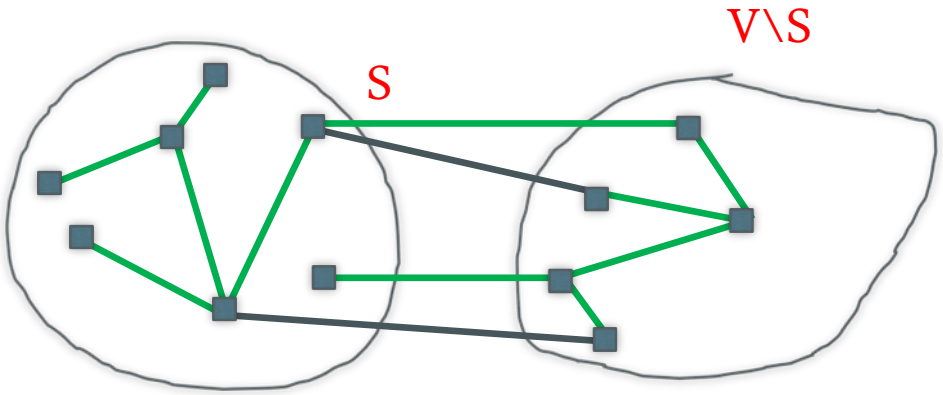
- **Not** a random contraction algorithm!
- $O(m \log^3 n)$ time randomized algorithm
- Based on *tree packings, Tutte-Nash Williams theorem* and *dynamic programming*

Key Definition

Cut $\delta(S)$

Spanning tree T

$\delta(S)$ 2-respects T if $|\delta(S) \cap E(T)| \leq 2$



Karger's mincut algorithm

1. Use randomized algorithm to find $O(\log n)$ trees $T_1, T_2, \ldots, T_{c \log n}$ such that $\delta(S^*)$ an optimum mincut 2-respects one of the trees with high probability
2. For each T_i
   - Simple algorithm that tries all possible pairs of edges of T_i runs in $O(n^2)$ time
3. Output cheapest cut found over all trees

Karger's mincut algorithm

1. Use randomized algorithm to find $O(\log n)$ trees T_1, T_2, …, $T_{c \log n}$ such that mincut $\delta(S^*)$ 2-respects one of the trees with high probability

2. For each T_i
   - Use clever dynamic programming algorithm to find smallest cut of G that 2-respects T_i in $O(m \log^2 n)$ time

3. Output cheapest cut found over all trees

Dynamic Mincut

[Thorup'01,'07]

Fully-dynamic algorithm for mincut

- $(1+\varepsilon)$ approximate
- $O(\sqrt{n})$ *amortized* update time for mincut value
- O(log n) per edge in mincut
- *Randomized* against oblivious adversary

Incremental Mincut

Question: how do we make Karger's algorithm *incremental* for MWU?

- weights only go up
- $(1+\varepsilon)$ approximation for mincut suffices

Reduction to Epochs

Start of epoch: compute mincut value λ

While mincut in G with weights w is $\leq (1+\varepsilon)\,\lambda$

Find *any* cut with capacity $(1+O(\varepsilon))\,\lambda$

Update weights lazily

End of epoch: mincut value $> (1+\varepsilon)\,\lambda$

Only $O(\log m/\varepsilon^2)$ epochs

Incremental Mincut

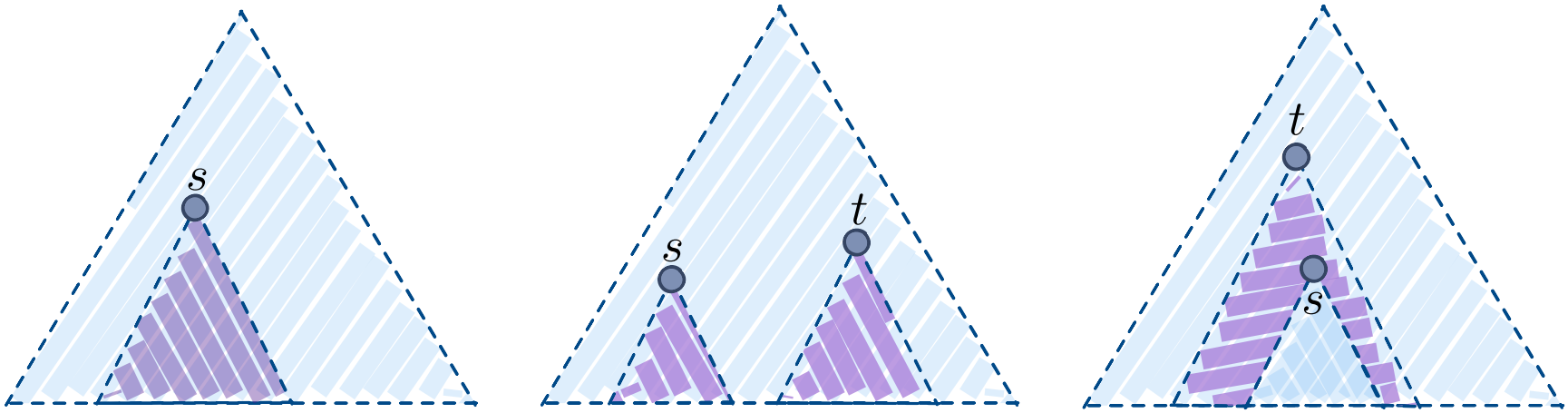
Question: how do we make Karger's algorithm *incremental*?

- weights only go up
- $(1+\varepsilon)$ approximation for mincut suffices

Solution: in each epoch run Karger's algorithm as if weights are not changing

(Modulo weight update to detect when mincut increases)

Types of cuts in a single tree



Preprocess each T_i at start of epoch

Use range tree data structure to store edges of G with edges of T for implicit access

Related Results for TSP

- Faster implementation of Christofides heuristic via LP relaxation
- Faster implementations of "Best of Christofides" heuristics
- s-t Path TSP LP and rounding

Sparsifying LP Solution

$$\min \sum_{e \in E} c_e x_e$$

$$\begin{aligned} x(\delta(S)) &\geq 2 & \emptyset \subset S \subset V \\ x_e &\geq 0 & e \in E \end{aligned}$$

Benczur-Karger cut sparsification can be applied to sparsify LP solution to get one with support $O(n \log n / \epsilon^2)$

Concluding Remarks

- MWU revisited for *implicit* LPs
  - Judicious use of algorithms/data structures can lead to substantial speed ups
  - Solving LP faster than combinatorial algorithms in some cases!
- Other problems, derandomization, better dependence on ε, parallel/distributed models, empirical evaluation, …

Thank You!