Speeding up MWU based Approximation Schemes and Some Applications

Chandra Chekuri
Univ. of Illinois, Urbana-Champaign

Joint work with Kent Quanrud
Based on recent papers

• *Near-linear-time approximation schemes for some implicit fractional packing problems* with Kent Quanrud, SODA 2017

• *Approximating the Held-Karp Bound for Metric TSP in Nearly-Linear Time* with Kent Quanrud, FOCS 2017

• *Randomized MWU for Positive Linear Programs* with Kent Quanrud, SODA 2018

• *Ongoing work*
Fast Algorithms

Recent trends/techniques

• Fast solvers for graph Laplacians [Spielman-Teng]

• Continuous/convex optimization methods: cutting planes, interior point, accelerated gradient descent …

• Sparsification (graphs, matrices, …)

• Sketching/streaming/embedding techniques for speeding up numerical linear algebra

• Data structures

• …
Bridging Continuous and Discrete Optimization
Multiplicative Weight Updates (MWU)

- Technique with many applications in TCS (see survey of [Arora-Hazan-Kale])
- Successfully used for fast FPTASes for LPs and beyond: [Grigoriadis-Khachiyan, Plotkin-Shmoys-Tardos] 1990’s
- Many subsequent ideas [Young, Luby-Nisan, Garg-Konneman, Garg-Khandekar, Fleischer, ...]
Our Work

• Revisit MWU based ideas for *implicit* problems in graphs, geometry etc
• Significantly faster algorithms for basic problems
• Initial work inspired by applications to *submodular optimization* [C-Jayram-Vondrak 2015]
(Pure) Packing LP

\[
\begin{align*}
\text{max } & \langle v, x \rangle \\
\text{subject to } & Ax \leq 1 \\
& x \geq 0
\end{align*}
\]

- \( n \): dimension of problem (size of \( x \))
- \( m \): number of rows of \( A \) (non-trivial constraints)
- \( N \): number of non-zeroes in \( A \)
- \( v, A \) non-negative
(Pure) Covering LP

\[
\begin{align*}
\min & \langle v, x \rangle \\
A x & \geq 1 \\
x & \geq 0
\end{align*}
\]

- \( n \): dimension of problem (size of \( x \))
- \( m \): number of rows of \( A \) (non-trivial constraints)
- \( N \): number of non-zeroes in \( A \)
Mixed Packing & Covering
aka Positive LP

\[ \text{A, B non-negative} \]

\[ \begin{align*}
    \mathbf{A} \mathbf{x} & \leq 1 \\
    \mathbf{B} \mathbf{x} & \geq 1 \\
    \mathbf{x} & \geq 0
\end{align*} \]

- **n**: dimension of problem (size of \( \mathbf{x} \))
- **m**: total number of rows in \( \mathbf{A} \) and \( \mathbf{B} \)
- **N**: total number of non-zeroes in \( \mathbf{A} \) and \( \mathbf{B} \)
Explicit Packing/Covering LPs

How fast can a \((1-\varepsilon)\) approximation be computed?

- MWU type algorithms:
  - randomized \(O(N + (n+m) \log n/\varepsilon^2)\) [Koufogiannakis-Young’07]
  - deterministic \(O(N \log n/\varepsilon^2)\) [Young’15]

- Accelerated gradient descent based algorithm:
  - randomized \(O(N \log n \log (1/\varepsilon)/\varepsilon)\) [AllenZhu-Orrechia’15, Wang-Mahoney-Rao’16]
Explicit Positive LPs

How fast can a \((1 - \varepsilon)\) approximation be computed?

- MWU based algorithms:
  - deterministic \(O(N \log n / \varepsilon^2)\) [Young’15]
Implicit Problems

- Matrices $A$ and $B$ defined implicitly by a combinatorial structure
- $N$ is “large” in terms of original input

This talk: focus on packing problems
Packing Spanning Trees

**Input:** graph $G=(V,E)$ edge capacities $c_e$

**Goal:** find a max fractional packing of spanning trees

\[
\max \sum_{T \in \mathcal{T}} x_T \\
\sum_{T \ni e} x_T \leq c_e \quad \forall e \in E \\
x_T \geq 0 \quad \forall T \in \mathcal{T}
\]
Interval Packing

- **n** closed intervals \( I_1, I_2, \ldots, I_n \): \( I_i = [a_i, b_i] \)
  - \( I_i \) has size \( d_i \) and value \( v_i \)

- **m** points \( p_1, p_2, \ldots, p_m \) on real line
  - \( p_j \) has capacity \( c_j \)
Interval Packing LP

\[
\text{max} \sum_{i=1}^{n} v_i x_i \\
\sum_{I_i \in p_j} d_i x_i \leq c_j \quad 1 \leq j \leq m \\
x_i \leq 1 \quad 1 \leq i \leq n \\
x_i \geq 0 \quad 1 \leq i \leq n
\]
TSP and Metric-TSP

- **TSP**: Undir graph $G=(V,E)$, edge costs $c_e$
  - find Hamiltonian Cycle in $G$ of minimum cost

- **Metric-TSP**: Undir graph $G=(V,E)$, edge costs $c_e$
  - find spanning *tour* in $G$ of minimum cost
  - same as Hamiltonian Cycle in *metric completion* of $G$
Subtour Elimination LP for TSP

\[
\begin{align*}
\min & \sum_{e \in E} c_e x_e \\
& \quad \sum_{e \in E} c_e x_e \\
& x(\delta(v)) = 2 \quad v \in V \\
x(\delta(S)) \geq 2 \quad \emptyset \subset S \subset V \\
x_e \geq 0 \quad e \in E
\end{align*}
\]
2ECSS LP

\[
\min \sum_{e \in E} c_e x_e \\
\quad x(\delta(S)) \geq 2 \quad \emptyset \subset S \subset V \\
\quad x_e \geq 0 \quad e \in E
\]

For Metric-TSP solving 2ECSS LP is equivalent to solving Subtour LP
## Faster Algorithms

<table>
<thead>
<tr>
<th>Problem</th>
<th>Previous best</th>
<th>New bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packing Spanning Trees</td>
<td>$O(mn \text{ polylog } n/\varepsilon^2)$</td>
<td>$O(m \text{ polylog } n/\varepsilon^2)$</td>
</tr>
<tr>
<td>Interval Packing LP</td>
<td>$O(mn \text{ polylog } /\varepsilon)$</td>
<td>$O((m+n) \text{ polylog } n/\varepsilon^2)$</td>
</tr>
<tr>
<td>Metric-TSP LP</td>
<td>$O(m^2 \text{ log } n/\varepsilon^2)$</td>
<td>$O(m \text{ polylog } n/\varepsilon^2)$ (randomized)</td>
</tr>
</tbody>
</table>

Ideas extend to positive LPs

Several other applications to LPs for combinatorial problems
Explicit Positive LPs

How fast can a \((1-\varepsilon)\) approximation be computed?

- MWU based algorithms:
  - deterministic \(O(N \log n/\varepsilon^2)\) [Young’15]

- [C-Quanrud’18] Randomized MWU
  - \(O(N \log n/\varepsilon + m/\varepsilon^2 + n/\varepsilon^3)\)
  - Randomization helpful in some implicit problems in computational geometry [C-HarPeled-Quanrud’18]
High-level Ideas

• Speed up classical MWU based approximation schemes

• Problem-specific *integration* of *dynamic data structures* for two separate issues
  • oracle for MWU
  • lazy weight update
Packing Spanning Trees
Packing Spanning Trees

**Input:** graph \( G=(V,E) \) edge capacities \( c_e \)

**Goal:** find a max fractional packing of spanning trees

\[
\begin{align*}
\max \ & \sum_{T \in \mathcal{T}} x_T \\
\sum_{T \ni e} x_T & \leq c_e \quad \forall e \in E \\
x_T & \geq 0 \quad \forall T \in \mathcal{T}
\end{align*}
\]
MWU Approach

- Maintain positive weights for constraints: $w_e$ for each edge $e$
- Start with $x = 0$, $w_e = 1/c_e$ for each edge $e$
- In each iteration solve Lagrangean relaxation: collapse $m$ constraints into 1 constraint
- \[
\max \sum y_T \quad \text{s.t.} \quad y \geq 0 \quad \text{and} \quad \sum_{e} w_e \sum_{e \in T} y_T \leq \sum_{e} w_e c_e
\]
- Take *small step* in direction of new solution $x = x + \delta y$
- Update weights exponentially in load
- Iterate for “one unit of time”
Solving Lagrangian Relaxation

In each iteration solve Lagrangian relaxation: collapse \( m \) constraints into 1 constraint

\[
\max \sum y_T \quad s.t. \quad y \geq 0 \quad and \quad \sum w_e \sum_{e \in T} y_T \leq \sum w_e c_e
\]

Rewriting:

\[
\max \sum y_T \quad s.t. \quad y \geq 0 \quad and \quad \sum_{T} w_T y_T \leq \sum w_e c_e
\]

Optimum solution: \( T^* \) that is MST wrt to weights \( w_e \)
How many iterations?

- In each iteration: $x = x + \delta \, y$
- $y$ is solution to relaxation and hence
  $$\sum_T y_T \geq OPT$$
- Run algorithm until time $= \sum \delta = 1$
- Guarantees optimality of final solution $x$
Weight and Load

Given current solution $x$

- Utilization of $e$, $u_e = \sum_{T \in e} x_T$
- Load on $e$, $\ell_e = u_e / c_e$

Maintain weight exponential in load

$$w_e = \frac{1}{c_e} \exp(\eta \ell_e)$$
Step size

Take small step: $x = x + \delta \quad y = x + \delta' 1_T$

How much should we add?

Want to change weights slowly:

$w'_e \leq (1 + \varepsilon)w_e \approx \exp(\varepsilon) w_e$

Implies (max) step size

$$\delta' = \frac{\varepsilon \min_{\eta \in T} c_f}{\eta \min_{f \in T} c_f}$$
Potential Function

Potential function: $\sum_e w_e c_e$

$m \leq \sum w_e c_e \leq m \exp(\eta \text{ time}) \leq m^{1+\eta}$


**Constraint Violation**

\[ w_e = \frac{1}{c_e} \exp(\eta \ell_e) \]

At end of algorithm

\[ \ell_e = \frac{1}{\eta} \ln(w_e c_e) \leq \frac{1}{\eta} \ln\left( \sum_e w_e c_e \right) \leq 1 + \frac{1}{\eta} \ln m \leq 1 + \epsilon \]

for \( \eta = \frac{1}{\epsilon} \ln m \)
Potential Function

Potential function: \( \sum_e w_e c_e \)

\[
m \leq \sum w_e c_e \leq m^{o\left(\frac{1}{\varepsilon}\right)}
\]

- In each iteration *at least* one edge weight increases by a multiplicative \((1 + \varepsilon) \approx \exp(\varepsilon)\) factor
- No edge’s weight updated more than \(O(\log \frac{m}{\varepsilon^2})\) times
- Total number of iterations is \(O(m \log \frac{m}{\varepsilon^2})\). *Width independent*
Run-time Analysis

- Number of iterations is $O(m \log m/\varepsilon^2)$
- In each iteration $h$ compute MST $T_h$ wrt current edge weights
- Update weights of edges in $T_h$
- Runtime:
  - $O(m \log m/\varepsilon^2 (m + n)) = O(m^2 \log m/\varepsilon^2)$
Improving run time via data structures

- Do not compute MST from scratch: maintain MST via dynamic data structure
  - Maintain weights lazily, update only if weight increases by \((1 + \varepsilon)\) factor – MST is approximate
  - Total number of updates is \(O(m \log m/\varepsilon^2)\)
  - MST update time per edge weight change is \(\text{polylog}(n)\)


\[
O(m \log m/\varepsilon^2 (\text{polylog}(n) + n)) = O(mn \text{polylog}(n)/\varepsilon^2)
\]

Bottleneck is weight update!
Updating weights in each iteration

\[ \exp(\epsilon) \]
\[ \exp(\epsilon/2) \]
\[ \exp(\epsilon/64) \]
\[ \exp(\epsilon/256) \]

\[ w \]

\[ A \]

\[ T \]

\[ c \]

\[ E \]
Updating weights lazily

- Can update weight lazily if it does not change much
  - maintain within a \((1 \pm \varepsilon)\) multiplicative actor

- When tree \(T_j\) is updated rate of change of \(w_e\) depends on \(c_e\)
  - if all \(c_e\) values are \(uniform\) then \(n-1\) edges increase weight by \((1+\varepsilon)\) factor
  - if \(c_e\) values are \(non-uniform\) delay updating small weight changes. \textbf{How?}
Borrow ideas from [Koufagiannis-Young’07, Young’15] for explicit problems

- **Deterministically**: Bucket $c_e$ values geometrically and lazily update using amortization. Somewhat involved to describe. Take advantage of $\{0,1\}$ structure of $A$

- **Randomization**: touch/update $w_e$ in proportion to $1/c_e$. 

Randomized weight update

When tree $T$ is the tree added in iteration $h$

- $c_{\text{min}}$ min edge capacity in $T$
- Pick random $\theta \in [0,1]$
- For each edge $e$ in $T$
  - If $\theta \leq c_{\text{min}}/c_e$ then $w_e = \exp(\epsilon) w_e$
  - Else $w_e$ remains the same
  - In expectation update is right
Updating weights in each iteration

\[ w_{t+1} = w_t \exp(\epsilon / 256) \]

\[ A \]

\[ T \]

\[ c \]

\[ \exp(\epsilon / 2) \]

\[ \exp(\epsilon / 64) \]

\[ \exp(\epsilon) \]

Multiplicative increase
Randomized weight update

• Pick random $\theta \in [0,1]$

• For each edge $e$ in $T$
  • If $\theta \leq c_{min}/c_e$ then $w_e = \exp(\epsilon) w_e$
  • Else $w_e$ remains the same

• Updates are “correlated” and hence easy to implement via data structures

• Catch: need to prove that MWU analysis works with correlated rounding - drift analysis
  [KY’07, CQ’18]
Packing Spanning Trees: Related Results and Applications

- Can approximately decompose a point in a spanning tree polytope into a convex combination of spanning trees in near-linear time
- Compact representation of a \((1+\varepsilon)\) packing with \(O(m\ \text{polylog } n/\varepsilon^2)\) edges
- Network strength and fractional packing # in \(O((m + n/\varepsilon^4)\text{polylog})\) time
- Applications to TSP, k-Cut, …
Metric-TSP
2ECSS LP

\[
\min \sum_{e \in E} c_e x_e
\]

\[x(\delta(S)) \geq 2\]
\[x_e \geq 0\]
\[\emptyset \subset S \subset V\]
\[e \in E\]

For Metric-TSP solving 2ECSS LP is equivalent to solving Subtour LP
Dual of 2ECSS LP

\[
\begin{align*}
\max 2 & \sum_S y_S \\
\sum_{S: e \in \delta(S)} y_S & \leq c_e \quad e \in E \\
y_S & \geq 0 \quad S \subset V
\end{align*}
\]

Packing cuts into capacities
Separation oracle is global mincut
MWU Algorithms

- Each iteration requires solving a global mincut problem: randomized $O(m \log^3 n)$ alg [Karger’00]
- $O(m \log m/\epsilon^2)$ iterations
- Leads to $O(m^2 \log^4 m/\epsilon^2)$ randomized algorithm [Plotkin-Shmoys-Tardos’95]
- $O(m^2 \log^2 m/\epsilon^2)$ deterministic algorithm [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

Make Karger’s algorithm *incremental* in a fashion that is suitable for MWU

Thorup’s fully dynamic *randomized* mincut algorithm has $O(\sqrt{n})$ update time but not suitable for MWU
Karger’s mincut algorithm

**Not** the contraction algorithm!

- Given $G=(V,E)$ compute (randomly) a sparsifier $H=(V,E')$ s.t. that mincut of $H$ is $O(\log n)$ and cuts are preserved.
- Compute $k=O(\log n/\varepsilon^2)$ spanning trees $T_1,..,T_k$ that $(1-\varepsilon)$-approximate max tree packing in $H$.
- By Tutte-NashWilliams theorem there is an $i$ such that $T_i$ 2-respects a mincut of $G$.
- Clever DP on each tree $T_j$ to find smallest cut of $G$ that 2-respects $T_j$ in $O(m \log^2 n)$ time.
Christofides Heuristic for Metric-TSP

1.2 TSP in Directed Graphs

In this subsection, we consider TSP in directed graphs. As in undirected TSP, we need to relax the problem conditions to get any positive result. Again, allowing each vertex to be visited multiple times is equivalent to imposing the asymmetric triangle inequality $c(u, w) \leq c(u, v) + c(v, w)$ for all $u, v, w$. This is called the asymmetric TSP (ATSP) problem. We are given a directed graph $G = (V, A)$ with cost $c(a) > 0$ for each arc $a \in A$ and our goal is to find a closed walk visiting all vertices. Note that we are allowed to visit each vertex multiple times, as we are looking for a walk, not a cycle. For an example of a valid Hamiltonian walk, see Fig 3.

The MST-based heuristic for the undirected case has no meaningful generalization to the directed setting. This is because costs on edges are not symmetric. Hence, we need another approach. The Cycle Shrinking Algorithm repeatedly finds a min-cost cycle cover and shrinks cycles, combining the cycle covers found. Recall that a cycle cover is a collection of disjoint cycles covering all vertices. It is known that finding a minimum-cost cycle cover can be done in polynomial time (see...
Christofides Heuristic for Metric-TSP

3/2-approximation

- Compute MST $T$ of $G=(V,E)$
- $S$: odd degree vertices of $S$
- Find min-cost $S$-join in $G$ via min-cost matching
- $T+S$-join is Eulerian and connected

- Running time: [Gabow-Tarjan’91]
  - Explicit metric: $\tilde{O}(n^{2.5})$
  - Implicit metric: $\tilde{O}(mn + n^{2.5})$
Christofides Heuristic: Faster Implementation via LP

• **Compute** MST $T$ of $G=(V,E)$

• $S$: odd degree vertices of $S$

• **Solve LP** to find solution $x$

• **Sparsify** via [Benczur-Karger] to reduce support of $x$ to $O(n \log n)$ edges

• **Find min-cost** $S$-join in sparsified graph via reduction to min-cost matching: $O(n \log n)$ sized graph

• $T+S$-join is Eulerian and connected

• Use [Wolsey’80] for analysis wrt to LP
Christofides Heuristic: Faster Implementation via LP

- New (randomized) run-time for $3/2 + \varepsilon$ approx.
  - Explicit metric: $\tilde{O}(n^2/\varepsilon^2 + n^{1.5}/\varepsilon^3)$
  - Implicit metric: $\tilde{O}(m/\varepsilon^2 + n^{1.5}/\varepsilon^3)$

- Previous run-time for $3/2$ approx. [Gabow-Tarjan’91]
  - Explicit metric: $\tilde{O}(n^{2.5})$
  - Implicit metric: $\tilde{O}(mn + n^{2.5})$
Remarks and Open Problems

- Weight update not a bottleneck for many MWU implementations for implicit problems
- Judicious use of data structures can lead to substantial speed ups in implementation of MWU based algorithms
- More applications for implicit packing/covering/mixed packing covering problems
- $O(1/\epsilon)$ dependence in run time?
- Parallel/distributed algorithms for implicit problems. NC algorithm for global mincut?
Thank You!