**CS 671: Graph Streaming Algorithms and Lower Bounds** **Rutgers: Fall 2020**

# Lecture 10

November 10, 2020

*Instructor: Sepehr Assadi* *Scribe: Vishwas Bhargava*

## 1 Introduction

In this lecture we will study an approach for solving transshipment problem and Single Source Shortest Path(SSSP) problem. This approach basically adapts the straight-forward LP formalization of the problem, but finds the optimal solution *smartly* so that it can be used in various other models like semi-streaming(multi-pass). The smart approach is to reach for the optimal solution via Gradient decent.

In the transshipment problem, we seek to find a cheapest routing for sending units of a single good from sources to sinks along the edges of a graph meeting the nodes demands. Equivalently, we want to find the minimum-cost flow in a graph where edges have unlimited capacity. The special case of SSSP can be modeled as a transshipment problem by setting the demand of the source to -n + 1 (thus supplying -n + 1 units) and the demand of every other node to 1. Unfortunately, this relation breaks when we consider approximation schemes: A $(1+\epsilon)$-approximate solution to the transshipment problem merely yields $(1+\epsilon)$-approximations to the distances on average. In the special case of SSSP, however, one is interested in obtaining a $(1+\epsilon)$-approximation to the distance for each single node and we show how to extend our algorithm to provide such a guarantee as well.

**Problem 1** (transshipment Problem)**.** Consider a bidirected graph $G = (V, E, w)$ where $V$ is the set of $n$ nodes, $E$ is the set of directed arcs, and $w$ assigns a positive integer weight $w_{(u,v)}$ to every arc $(u, v) \in E$. Let $W$ denote the $2m \times 2m$ diagonal matrix containing the forward and backward weights and let $\mathbf{b} \neq 0 \in \mathbb{Z}^n$ be a vector of demands s.t. $\mathbf{b}^T\mathbf{1} = 0$, that is the positive demands equal the negative demands a.k.a supplies. Also, let A be the incidence matrix of a directed graph contains a row for every node and a column for every arc and $A_{i,j}$ is 1 if the j-th arc enters the i-th node, $-1$ if it leaves the node, and 0 otherwise. As an exercise, observe that $A^T\mathbb{1} = 0$.

The transshipment problem can then be written as a primal/dual pair of linear programs:

$$min\{\mathbf{1}W\mathbf{x} : A\mathbf{x} = b, \mathbf{x} \geq 0\} = max\{\mathbf{b}^T\mathbf{y} : (W^{-1}A^T\mathbf{y})_{max} \leq 1\}. \tag{1}$$

The dual (right) program asks for node potentials y such that for each $arc(u, v) \in E$, $y_uy_v \leq w_(u, v)$, maximizing $\mathbf{b}^T\mathbf{y}$. Note that, because $\mathbf{b}^T\mathbf{1} = \mathbf{0}$, shifting the potential by $\mathbf{r}\cdot\mathbf{1}$ for any $r \in R$ does neither change $\mathbf{b}^T\mathbf{y}$ nor $y_u - y_v$ for any $u, v \in V$. The goal of the dual is thus to maximize the differences in potential of sources and sinks (weighted according to b), subject to the constraint that the potential of u does not exceed the potential of v by more than $w_{(u,v)}$ for any neighbor $v$ of $u$.

Another related problem to the transshipment problem and also very important on its own is the problem of single source shortest path(SSSP). The aim here is to find shortest path from a given note (source s) to all other vertices of the graph. Note that, SSSP is a special instance of transshipment problem where source node has demand $-(n-1)$ and every other node has demand $+1$. The approximate version of SSSP asks us to approximate each optimal path from source to other other vertices. Note that, running approximate version of transshipment algorithm on special case of SSSP will *not* guarantee an approximate solution to SSSP. It will just approximate upper bounds on the distances from the source s on average. The two main results discussed in this lecture are the following:

1. A polynomial time $O(\log n)$-pass streaming algorithm for $(1+\epsilon)$-approximation of Transshipment problem.
2. A polynomial time $O(\log n)$-pass streaming algorithm for $(1+\epsilon)$-approximation of Single source shortest path problem.

We will now describe Gradient descent basic principle which we will use later to give good semi-streaming algorithm for transshipment/SSSP problems that computes a near-optimal solution.

## 1.1 An Efficient Oracle for O($\log n$)-Approximate Solutions

In this section we will show that we can simply compute an optimal solution to (1) on an $\alpha$-spanner which in-turn will be $O(\alpha)$ approximate to optimal solution of (1). Think of $\alpha$ as $O(\log n)$, note that we can compute an optimal solution to (1) on $O(\alpha)$ spanner just by saving the full spanner in our memory and apply LP based methods. The next lemma proves the aforementioned claim.

**Lemma 1.** *Given an $\alpha$-spanner $S$ of $G$, let $x$ and $y$ be optimal solutions to (1) on the spanner. Then $\alpha^{-1}y$ is feasible on $G$ with objective is by factor $\alpha$ smaller than that of $y$.*

*Proof.* Let $\{u,v\} = e \in E$ be an edge in $G$. By the definition of a spanner, there is a path $P_{uv}$ of weight at most $\alpha w_e$ from $u$ to $v$ in S. We have,

$$\begin{aligned}
|(W^{-1}A^Ty)_(u,v)| &= \frac{|y_u - y_v|}{w_e} \\
&\leq \frac{\sum_{(u',v')\in P_{uv}} |y_{u'} - y_{v'}|}{w_e} && \text{Telescoping via path and applying } \Delta\text{-inequality} \\
&\leq \frac{\sum_{(u',v')\in P_{uv}} |y_{u'} - y_{v'}|}{w_e} \\
&= \frac{\sum_{(u',v')=e'\in P_{uv}} w'_e |(w_{e'}^{-1}A^Ty)_(u,v)|}{w_e} \\
&\leq \frac{\sum_{(u',v')=e'\in P_{uv}} w'_e}{w_e} \leq \alpha
\end{aligned}$$

Thus, $\alpha^{-1}y$ is feasible on G. □

Thus, an $\alpha$-approximate primal-dual pair of solutions to (1) with demands b can be computed (without further knowledge of G).

## 1.2 Generic Gradient Descent Algorithm

```
1. Pick a differentiable potential function that reflects your objective.
2. Find a good starting point.
3. **While**: You are *not* sufficiently close to the optimum solution.
    (a) Determine the gradient of the potential function at the current solution.
    (b) Determine a direction for the update in which the gradient indicates that the potential reduces
        quickly.
    (c) Choose a large step width, under the constraint that the gradient does not change too much
        along the way, and update the solution according to direction and step width.
```

Note that our LP formulation doesn't directly yield a good objective function as both the primal and dual program in (1) have constraints. Thus, we can't start apply any gradient descent based approach on these objective functions. We will instead work with a slightly different objective function which has just one constraint instead of $O(n)$ in(1) and turns out to be equivalent and the equivalence continue to hold in the approximate setting as well.

$$min\{(W^{-1}A^T\pi)_{max} : b^T\pi = 1\}. \tag{2}$$

**Lemma 2.** *1. If $\pi$ is a feasible solution of (2) then $\psi(\pi) := \frac{\pi}{(W^{-1}A^T\pi)_{max}}$ defines a feasible solution of the dual in (1). If $y$ is a feasible dual solution of(1) satisfying $b^Ty > 0$, then $\chi(y) := \frac{y}{b^Ty}$ defines a feasible solution of (2).*

*2. The map $\psi(\cdot)$ preserves the approximation ratio. Namely, for any $\gamma \geq 1$, if $\pi$ is a solution of (2) within factor $\gamma$ of the optimum, i.e. $(W^{-1}A^T\pi)_{max} \leq \gamma(W^{-1}A^T\pi^*)_{max}$ then $\psi(\pi)$ is feasible for (1) and within factor $\gamma$ of the optimum, i.e. $b^T\psi(\pi) \geq \gamma^{-1}b^ty^*$.*

*Proof.* See [1] lemma 2.5 □

In other words, it is sufficient to determine a $(1+\epsilon)$-approximation to (2) in order to obtain a $(1+\epsilon)$-approximation to (1).

We will apply the adopt the following strategy to maintain the constraint of $b^T\pi = 1$: Our initial solution $\pi$ will satisfy $b^T\pi = 1$ and we will ensure our updates $\pi' \leftarrow \pi - h$ satisfy $b^Th = 0$, so overall $b^T(\pi') = b^T(\pi) - b^T(h) = 1$. Also, note that feasible primal solutions of (2) on a spanner are still feasible on G. Hence the same arguments as before yield that an $\alpha$-approximate pair can be computed based on an $\alpha$-spanner of G.

Now, we have ensured that we have a good objective function with just one constraint(and a strategy to deal with that constraint) but we are still not done, as our potential function is not differentiable! Indeed, *max* of a bunch of linear functions is not differentiable(but it is continuous nonetheless). In the next section we use some standard approach of smoothing the gradient function to overcome this obstacle.

## 1.3 Making our objective function "smooth"

For the maximum value of a vector, a suitable candidate is given by the **log-sum-exponent** function.

**Definition 3** (log-sum-exponent)**.** For any vectors $z \in \mathbb{R}^d$ , it is defined as

$$lse_\beta(z) := \frac{1}{\beta}\ln\left(\Sigma_{i\in[d]}e^{\beta z_i}\right)$$

where the parameter $\beta > 0$ determines the trade-off between accuracy of approximation and smoothness.

Properties of $lse_\beta(z)$ that we will be needing in our anlysis,

1. $(\mathbf{z})_{max} \leq lse(\mathbf{z}) \leq (\mathbf{z})_{max} + \frac{\ln d}{\beta}$. Follows directly from definition.
2. $$\nabla lse_\beta(\mathbf{z}) - \nabla lse_\beta(\mathbf{z'})||_1 \leq \beta||\mathbf{z} - \mathbf{z}'||_\infty. \qquad [2] \tag{3}$$

Define new potential function,

$$\Phi_{\beta(\pi)} := lse_\beta(W^{-1}A^T\pi).$$

Note that $\Phi_\beta(\cdot)$ is convex for any $\beta$, as it is constructed by composing $lse_\beta()$, which is convex for any $\beta$ with linear functions. To show that $\Phi_\beta(\pi)$ differentiable, we examine the gradient of the potential function. By chain rule and since multiplication with matrices $W^{-1}$ and $A^T$ is a linear function, we get that

$$\nabla\Phi_\beta(\pi) = AW^{-1}\nabla lse_\beta\left(W^{-1}A^T\pi\right) \tag{4}$$

Now, we will show that $\Phi_\beta(\pi)$ is differentiable or equivalently, $\nabla\Phi_{\beta(\pi)}$ is continuous.
Concretely we will show that,

**Claim 4.**

$$\forall\pi, h \in \mathbb{R}^n, |\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h| \leq \beta(W^{-1}A^T h)^2_{max}. \tag{5}$$

*Proof.*

$$\begin{aligned}
\pi, h \in \mathbb{R}^n : & |\nabla\Phi_\beta(\pi)^T h - \nabla\Phi_\beta(\pi - h)^T h| \\
&= \left|AW^{-1}\nabla lse_\beta\left(W^{-1}A^T\pi\right)^T h - AW^{-1}\nabla lse_\beta\left(W^{-1}A^T(\pi - h)\right)^T h\right| && \textit{by } (4) \\
&= \left|\left(\nabla lse_\beta\left(W^{-1}A^T\pi\right) - \nabla lse_\beta\left((\pi - h)\right)\right)^T \cdot W^{-1}A^T h\right| \\
&= \left|\left|\left(\nabla lse_\beta\left(W^{-1}A^T\pi\right) - \nabla lse_\beta\left((\pi - h)\right)\right)^T\right|\right|_1 ||W^{-1}A^T h||_\infty && \text{Holder} \\
&= \left|\left|\left(\nabla lse_\beta\left(W^{-1}A^T\pi\right) - \nabla lse_\beta\left((\pi - h)\right)\right)^T\right|\right|_1 ||W^{-1}A^T h||_\infty && \textit{by } (3) \\
&= \beta(W^{-1}A^T h)^2_{max}.
\end{aligned}$$

□

Now that our objective function is differentiable we will focus our concern to the approximation guarantee, our target of a $(1+\epsilon)$-approximation entails that the potential must be a more accurate approximation to the objective. Accordingly, we will ensure that

$$\Phi_\beta(\pi) \leq \left(1 + \frac{\epsilon}{4}\right)(W^{-1}A^T\pi)_{max}. \tag{6}$$

By point 1 of properties of lse, ensuring this condition suffices, $4\ln(2m) \leq \epsilon\beta(W^{-1}A^T\pi)_{max}$.

## 2 Algorithm for Transhipment problem

Our algorithm will start with initial $O(\log n)$ approximate solution to (2). Our next lemma shows how does the our approximation improves, in other words our potential decreases, when we proceed in our gradient decent algorithm. That is, how does $\phi_\beta(\pi)$ changes when we change $\pi$ to $\pi - h$.

**Lemma 5.** *Suppose $\pi, h \in \mathbb{R}^n$ satisfy, $\nabla\Phi_\beta(\pi)^T h > 0$ and $(W^{-1}A^T h)_{max} \leq 1$. Then for $\delta := \nabla\Phi_\beta(\pi)^T h$ it holds that*

$$\Phi_\beta\left(\pi - \frac{\delta h}{2\beta}\right) \leq \Phi_\beta(\pi) - \frac{\delta^2}{4\beta}.$$

*Proof.* Let us denote $\tilde{h} := \frac{\delta h}{2\beta}$. Recall that $\Phi_\beta(\cdot)$ is convex and thus $\Phi_\beta(\pi) \geq \Phi_\beta(\pi - \tilde{h}) + \nabla\Phi_\beta(\pi - \tilde{h})^T\tilde{h}$. This gives

$$\begin{aligned}
\Phi_\beta(\pi - \tilde{h}) - \Phi_\beta(\pi) &\leq \nabla\Phi_\beta(\pi - \tilde{h})^T\tilde{h} + \Phi_\beta(\pi)^T\tilde{h} - \Phi_\beta(\pi)^T\tilde{h} \\
&\leq \beta(W^{-1}A^T\tilde{h})^2_{max} - \Phi_\beta(\pi)^T\tilde{h} && \textit{Claim } 1.3 \\
&\leq -\frac{\delta^2}{4\beta}.
\end{aligned}$$

□

As this lemma suggests, we will try to ensure large progress by making $\delta = \nabla\Phi_\beta(\pi)^T h$ as large as possible and $\beta$ as small as possible. Now observe that, for a fixed $\beta$, maximizing $\delta$ under the constraint $(W^{-1}A^T h)_{max} \leq 1$ is another instance of the transshipment problem with demand vector $\nabla\Phi_\beta(\pi)$. Our algorithm will determine its step direction $h$ by computing an $\alpha$-approximation to this transshipment instance. Regarding the incentive of minimizing $\beta$, note that progress with respect to $\Phi_\beta$ is meaningless if it does not provide a sufficiently accurate approximation of the true objective $(W^{-1}A^T(\cdot))_{max}$ , so we will increase $\beta$ only when it becomes necessary to ensure (6). Bounding $\beta$ from above turns the additive progress guarantee into a relative one.

**Corollary 6.** *Suppose $\pi, h \in \mathbb{R}^n$ satisfy $\nabla\Phi_\beta(\pi)^T h > 0$, $(W^{-1}A^T h)_{max} \leq 1$, and $\epsilon\beta\Phi_\beta(\pi) \leq 10\ln 2m$. Then for $\delta = \nabla\Phi_\beta(\pi)^T h$ it holds that*

$$\Phi_\beta\left(\pi - \frac{\delta h}{2\beta}\right) \leq \left(1 - \frac{\epsilon\delta^2}{40\ln(2m)}\right)\Phi_\beta(\pi).$$

## 2.1 How to minimize the number of iterations

As mentioned before reducing the number of iterations is linked with determining a direction for the update in which the gradient indicates that the potential reduces quickly. And that this boils down to solving another instance of the transshipment problem with demand vector $\Phi_\beta(\pi)$. (Exercise: verify that for this demand vector demand = supply).

We now present the pseudocode of the generic gradient descent algorithm for a $(1+\epsilon)$-approximate solution to (1).

```
Algorithm: Gradient Transshipment($G, b, \epsilon$).
  1. compute $\alpha$-approximation $\pi$ to (2).
  2. While $\delta > \frac{\epsilon}{6\alpha}$:
      (a) Compute $\nabla\Phi_\beta(\pi)$.
      (b) Compute an $\alpha$-approximate solution $h$ to $max\{\nabla\Phi_\beta(\pi)^T\tilde{h} : (W^{-1}A^T\tilde{h})_{max} \leq 1) \wedge b^T\tilde{h} = 0\}$.
      (c) Set $\delta := \nabla\Phi_\beta(\pi)^T h$.
      (d) $\pi \leftarrow \pi - \frac{\delta h}{2\beta}$.
      (e) While: $\beta < \frac{4\ln(2m)}{\epsilon(W^{-1}A^T\pi)_{max}}$ do $\beta \leftarrow 2\beta$
  Output: $\pi$, dual $(1+\epsilon)$-approximate solution of (1).
```

There are two steps which needs explanations. Firstly, why do we stop when $\delta = \nabla\Phi_\beta^T h$ becomes small enough. This is due to the fact that $\nabla\Phi_\beta^T h = 0$ at minima, so when $\delta$ is sufficiently small this means that we are already close enough to the optimal solution. Secondly, why are we doubling $\beta$? To ensure smoothness and good approximation to our objective, we have maintained an invariant that $4\ln 2m \leq \epsilon\beta(W^{-1}A^T\pi)_{max} \leq \epsilon\beta\Phi_\beta(\pi) \leq 10\ln 2m$. However, when we update $\pi$ in step (2d) that might violate this, thus we increase $\beta$.

Now, we will bound the number of iterations, and later discuss how to discuss each iteration efficiently in *streaming* setting.

Recall, aim is to minimize potential: $\Phi_\beta(\pi) = lse(W^{-1}A^T\pi)$.

- If inner loop doesn't terminate, then potential decreases by a factor of $\left(1 - \frac{\epsilon_i\delta^2}{40\ln(2m)}\right)$.
- Potential can increase when we double $\beta$. The potential can increase by a factor of atmost $\left(1 + \frac{\epsilon_i}{4}\right)$. Also, this won't happen when $i > 1$ and when $i = 1$, number of times $\beta$ doubles is bounded by $\log\alpha$.

- Using above two insights, we can bound the number of iterations by $O\left((\epsilon^{-2} + \log\alpha)\alpha^2 \log n\right)$. For details, see [1] Lemma 2.13

**Remark.** Although we have given some intuitive reason why the output of the aforementioned algorithm is a $(1+\epsilon)$-approximation algorithm its not a formal proof. See lemma 2.11 of [1]. Also, note that we just output a dual solution instead of primal. The authors of [1] overcome this and can compute primal solution as well see observation 2.8 and 2.9.

## 2.2 Efficient implementation in *streaming* setting

Note that the only two steps which require adaptation in streaming setting are computing $(W^{-1}A^T\pi)_{max}$ and $\nabla\Phi_\beta(\pi)$. To implement these, define $s_e(\pi) := \frac{\pi_u - \pi_v}{w(e)}$ for every arc $e \in E$, both $\sum_{e\in E} e^{\beta s_e(\pi)}$ and $(W^{-1}A^T\pi)_{max} = max\{s_e(\pi) : e = (u,v) \in E\}$ can be computed in a single pass with $O(1)$ temporary space as summation and maximum are associative and commutative operators: Before the pass, we create temporary variables $\mathsf{s}$ and $\mathsf{m}$, both initialized to 0. During the pass, every time $w_e$ read an arc $e$ of weight $w_e$ from the stream, we first compute $s_e(\pi)$ and then update $\mathsf{s}$ to $\mathsf{s}$ $+e^{\beta s_e(\pi)}$ and $\mathsf{m}$ to $\max\{\mathsf{m}, s_e(\pi)\}$. After the pass, we have $\mathsf{s} = \sum_{e\in E} e^{\beta s_e(\pi)}$ and $\mathsf{m} = (W^{-1}A^T\pi)_{max}$. Since each component $\nabla\phi_\beta(\pi)$ is given by

$$\begin{aligned}\nabla\phi_\beta(\pi)_v &= (AW^{-1}\nabla lse_\beta(W^{-1}A^T\pi))_v \\ &= \sum_{e=(u,v)\in E} \frac{e^{\beta s_e(\pi)}}{w_e \cdot \sum_{e'\in E} e^{\beta s_{e'}(\pi)}} - \sum_{e=(v,u)\in E} \frac{e^{\beta s_e(\pi)}}{w_e \cdot \sum_{e'\in E} e^{\beta s_{e'}(\pi)}}\end{aligned}$$

the same idea can be used to compute $\nabla\Phi_\beta(\pi)$ in a single pass with O(1) temporary space, once $\mathsf{s} = \sum_{e\in E} e^{\beta s_e(\pi)}$ is known.

# 3 Single-Source Shortest Paths

We start this section with the following lemma.

**Lemma 7.** *(1) has an optimal primal solution that sends flow only along the arcs of a forest.*

What this means is that optimal primal solution is respresentable in O(n) space, justying that we can infact output/store the solution in streaming setting. For a proof of this statement see Lemma 3.1 of [1].

As discussed in the introduction, SSSP is a special case of Transshipment problem with single source node with negative demand and the demand on every non-source node is either 0 or 1. However, there are two issues with using the transshipment algorithm directly. Firstly, if a primal solution is computed, there is no guarantee that it induces a tree, and thus it is not clear which arc one should traverse from v when searching for a short path to s. Secondly, approximating the optimal solution guarantees only that the computed upper bounds on the distances from the source s are a $(1+\epsilon)$-approximation on average. We will discuss how to fix these issues below.

## 3.1 Sampling Tree solution

We construct a (primal) tree solution that is good on average. We assume in the following that there is a single source node with negative demand and the demand on every non-source node is either 0 or 1. The idea, which relies on our specific choice of a spanner-based oracle, is as follows:

1. Run Algorithm for transshipment problem, let $x$ the returned primal solution.

2. Partition its incoming arcs into classes in which arc weights differ by factor at most 2.

3. Sample (u, v) with probability $f_{(u,v)}^{-1}(2\alpha+1)x_{(u,v)}$, where $f_{(u,v)}$ denotes the sum of flows of arcs in the class of (u, v).
4. We can get $(1+\epsilon)$-approximation of an optimal solution using only sampled arcs and spanner edges is a with high probability.
5. we can bound the number of arcs sampled by the procedure by $O(\alpha n \log n)$ with high probability.

In order to prove the correctness of the above procedure, conceptually decompose $x =: x^{\Delta} + x^{o}$, where the arcs with non-zero flow in $x^{\Delta}$ form a directed acyclic graph (DAG). For details and proof of correctness see section 3.1 of [1].

## 3.2 Computing an Approximate Shortest-Path Tree

The approach here is simple:

1. Solve a single-source transshipment instance.
2. Mark nodes which have found a suitable approximate shortest path in the current tree as ”done”.
3. Set demand of done node to zero and repeat.

Again, for details and proof of correctness see section 3.2 of [1].

## References

[1] R. Becker, A. Karrenbauer, S. Krinninger, and C. Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*, pages 7:1–7:16, 2017. 3, 6, 7

[2] J. Sherman. Nearly maximum flows in nearly linear time. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 263–269. IEEE Computer Society, 2013. 3