COMPSCI 638: Graph Algorithms September 6, 2019

Lecture 4

Lecturer: Debmalya Panigrahi *Scribe: Kevin Sun*

1 Overview

In this lecture, we introduce a generalized version of the maximum flow problem known as the multicommodity flow problem. To solve this problem, we present and analyze an algorithm given by Garg and Könemann [GK07]. This algorithm illustrates a general class of algorithms known as primal-dual algorithms.

2 Multicommodity Flow

The multicommodity flow problem generalizes the maximum flow problem to multiple source-sink pairs. We are given a directed $G = (V, E)$ with edge capacities $u(e)$, as well as (s_i, t_i) pairs for $i \in \{1, \dots, k\}$ for some $k \geq 1$. Our objective is to find a feasible flow for every (s_i, t_i) pair that maximizes the total flow value in the network.

A False Reduction: Before we present today's algorithm, consider the following "reduction" of this problem to the maximum flow problem: add a "super" source x, a "super" sink y to the graph, and all edges of the form (x, s_i) and (t_i, y). In this new network, we can return a maximum x-y flow in the new network. The problem with this reduction is that it might send flow from s_1 to t_2, say, but any s_1-t_2 flow does not contribute to the actual objective value.

2.1 The LP and its Dual

Our algorithm for maximum multicommodity flow heavily relies on its LP formulation and its dual, so we present those now. Note that they closely resemble the primal and dual LPs for maximum flow presented in Lecture 2—we are simply extending the number of source-sink pairs to $k \geq 1$. We let P_i denote the set of s_i-t_i paths, and we let $P = \cup_i P_i$ denote the set of all such paths.

$$\text{(P): } \max \sum_{i=1}^{k} \sum_{p \in P_i} f_p$$

$$\sum_{i=1}^{k} \sum_{\substack{p \in P_i: \\ e \in p}} f_p \leq u(e) \quad \forall e \in E$$

$$f_p \geq 0 \quad \forall p \in P$$

And similarly, the dual of this LP is the following.

$$\text{(D): } \min \sum_{e \in E} u(e)\ell_e$$
$$\sum_{e \in p} \ell_e \geq 1 \quad \forall p \in P$$
$$\ell_e \geq 0 \quad \forall e \in E$$

Recall that we interpret the ℓ_e variables as edge lengths: in the dual LP, our objective is to minimize the total volume while ensuring that the distance from any source to its destination is at least one.

A polynomial-time algorithm: Before presenting the main algorithm of this lecture, we note obtaining that a polynomial-time algorithm is fairly simple. To do this, we'll need a *separation oracle* for (D): this is an algorithm that, given a candidate solution ℓ for (D), verifies that ℓ is feasible or returns a violated constraint. For our problem, this separation oracle computes the shortest path according to ℓ for every source-sink pair. If all k shortest paths have length at least one, then ℓ is feasible; otherwise, the shortest path gives a violated constraint.

With this separation oracle, we can use the ellipsoid algorithm for linear programming to solve the maximum multicommodity flow problem in polynomial time. Unfortunately, the runtime of this strategy is somewhat high, so we now turn our attention to a faster algorithm.

2.2 A Primal-Dual Algorithm

We now describe the algorithm for the maximum multicommodity flow problem given by Garg and Könemann [GK07]. Note that this is algoritm is actually an approximation scheme: as part of the input, we specify a constant $\epsilon \in (0, 1)$. The output of the algorithm is not necessarily a maximum flow, but as we will show, its value is guaranteed to be at least $(1 - \epsilon)$ times the maximum value.

Algorithm 1 (Garg and Könemann [GK07])

```
1:  Initialize: $f \leftarrow 0, \ell_e \leftarrow \delta \quad \forall e \in E$ (for some $\delta$ determined in the analysis).
2:  while $\ell$ is not feasible for the LP (D) do
3:      $p \leftarrow$ shortest path in $P$ under $\ell$
4:      $\gamma \leftarrow \min_{e \in p} u(e)$
5:      $g \leftarrow$ flow sending $\gamma$ units along $p$
6:      $f \leftarrow f + g$
7:      for $e \in p$ do
8:          $\ell_e \leftarrow \left(1 + \epsilon \frac{\gamma}{u(e)}\right) \ell_e$
9:      end for
10: end while
```

Now we shall prove three main properties of this algorithm: it terminates, the resulting flow is feasible, and the resulting flow value is at least $(1 - \epsilon)$ times the maximum flow value.

Lemma 1. *Algorithm 1 terminates after at most $m \log_{1+\epsilon} \left(\frac{1+\epsilon}{\delta}\right)$ iterations.*

Proof. Consider the value of ℓ_e for some fixed edge $e \in E$. Whenever e gets saturated (i.e., some augmenting flow sends $u(e)$ units of flow along e), the value of ℓ_e increases by a factor of $(1 + \epsilon)$.

The initial value of ℓ_e is δ, and the final value is at most $1+\epsilon$ because the algorithm only increases the ℓ_e values along the *shortest* path p. (If the shortest path length is at least $1+\epsilon$, then the algorithm would have terminated.)

Thus, the number of times an edge e can be saturated is at most the number of times δ can be multiplied by a factor of $(1+\epsilon)$ before it exceeds $1+\epsilon$, which is $\log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right)$.

Since each iteration of the algorithm saturates at least one edge, the total number of iterations is at most the total number of saturations, which is at most $m\log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right)$. ☐

To simplify the presentation of the remaining proofs, we make the following assumption:

$$(1+\epsilon)^{\Delta} = 1+\Delta\cdot\epsilon+\frac{\Delta(\Delta-1)}{2!}\epsilon^2+\cdots\approx 1+\Delta\cdot\epsilon, \tag{1}$$

where the equality follows from the Taylor series expansion of $f(x)=(1+x)^{\Delta}$ and the approximation follows from the fact that ϵ is a small constant.

Lemma 2. *Scaling the flow from Algorithm 1 by* $1/\log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right)$ *yields a feasible flow.*

Proof. Let e be some fixed edge and consider the value of ℓ_e. By the assumption (1), when we send γ units of flow through e, the value of ℓ_e increases by a factor of

$$\left(1+\epsilon\frac{\gamma}{u(e)}\right)\approx(1+\epsilon)^{\gamma/u(e)}.$$

Thus, if we send $f_e=\gamma_1+\gamma_2+\cdots+\gamma_j$ units of flow through e, then

$$\ell_e\approx\delta(1+\epsilon)^{\gamma_1/u(e)}\cdots(1+\epsilon)^{\gamma_j/u(e)}=\delta(1+\epsilon)^{f_e/u(e)}\le 1+\epsilon,$$

where the inequality again follows from the fact that the algorithm never extends an edge length beyond $1+\epsilon$. Rearranging the inequality above yields

$$f_e\le u_e\cdot\log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right),$$

so scaling the result by this factor indeed results in a feasible flow. ☐

So now we know that Algorithm 1 terminates with a feasible flow. The last thing we must do is relate the value of this flow to the value of the maximum flow by looking at the dual (D).

Theorem 3. *The value of the scaled flow given by Algorithm 1 and Lemma 2 is at least* $(1-\epsilon)$ *times the maximum flow value.*

Proof. We begin with some notation: for any length function $\ell: E\to\mathbb{R}_{\ge 0}$, we define $D(\ell)$ as the dual objective value obtained by ℓ and $\alpha(\ell)$ as the length of the shortest path in P under ℓ (i.e., the "most-violated" dual constraint value). Now observe that (D) is invariant under scaling: if we multiply ℓ by some fixed scalar value, then the values of $D(\ell)$ and $\alpha(\ell)$ are scaled by the same scalar value. Thus, we can solving (D) is equivalent to minimizing $D(\ell)/\alpha(\ell)$ over all functions $\ell: E\to\mathbb{R}_{\ge 0}$ without any constraints. Let β denote the optimal value of this reformulated problem.

Now let $\ell_0, \ell_1, \dots, \ell_t$ denote the length functions obtained when we run Algorithm 1, so that $\ell_0(e) = \delta$ for every $e \in E$ and ℓ_t is a feasible dual solution. Fix $i \in \{1, \dots, t\}$ and consider the length function $\ell_i - \ell_0$. Recall that β is the minimum value of $D(\ell)/\alpha(\ell)$ over *all* length functions, so

$$\beta \leq \frac{D(\ell_i - \ell_0)}{\alpha(\ell_i - \ell_0)} \leq \frac{D(\ell_i) - D(\ell_0)}{\alpha(\ell_i) - \delta m}. \tag{2}$$

Note that we have used the fact that D is a linear function, and $\alpha(\ell_i - \ell_0) \geq \alpha(\ell_i) - \delta m$ because reducing the length of every edge by δ cannot reduce the shortest path length by more than δm.

Consider the numerator of (2). For any iteration j, let p_j denote the path we chose in step j (so $\alpha(\ell_j)$ is the length of p_j under ℓ_j), and let γ_j denote the flow sent along p_j. Then we have

$$D(\ell_i) = D(\ell_0) + \sum_{j=0}^{i-1} \sum_{e \in p_j} \frac{\epsilon \gamma_j}{u(e)} \cdot u(e) \ell(e) = D(\ell_0) + \epsilon \sum_{j=0}^{i-1} \gamma_j \alpha(\ell_j). \tag{3}$$

Substituting (3) into (2) yields

$$\beta \leq \frac{\epsilon \sum_{j=0}^{i-1} \gamma_j \alpha(\ell_j)}{\alpha(\ell_i) - \delta m}. \tag{4}$$

Now we focus on the denominator. Rearranging (4) yields

$$\alpha(\ell_i) \leq \delta m + \frac{\epsilon}{\beta} \sum_{j=0}^{i-1} \gamma_j \alpha_j.$$

To obtain a closed-form bound on α_i, we assume the above inequality is tight (i.e., holds with equality) and solve the resulting recurrence. More formally, we define x_i so that the above inequality holds with equality, i.e.,

$$x_i = \delta m + \frac{\epsilon}{\beta} \sum_{j=0}^{i-1} \gamma_j x_j,$$

and we have $\alpha(\ell_i) \leq x_i$ and $x_0 = \delta m$. Now notice that we have a telescoping sum:

$$x_i - x_{i-1} = \frac{\epsilon}{\beta} \gamma_{i-1} x_{i-1},$$

and repeated applications of this equality gives us

$$\begin{aligned}
x_i &= \left(1 + \frac{\epsilon}{\beta} \gamma_{i-1}\right) x_{i-1} \\
&\approx (1+\epsilon)^{\gamma_{i-1}/\beta} x_{i-1} && \text{(by the assumption (1))} \\
&\approx (1+\epsilon)^{\gamma_{i-1}/\beta} (1+\epsilon)^{\gamma_{i-2}/\beta} x_{i-2} \\
&\vdots \\
&= (1+\epsilon)^{\sum_{j=0}^{i-1} \gamma_j/\beta} \delta m,
\end{aligned}$$

where the last line follows because $x_0 = \delta m$. Since ℓ_t is a feasible solution to (D), we have

$$1 \leq \alpha(\ell_t) \leq x_t = (1+\epsilon)^{|f|/\beta} \delta m$$

where $|f|$ denotes the flow value obtained by the algorithm *before scaling*. Once we account for the scaling factor from Lemma 2, we see that the approximation ratio of Algorithm 1 is at least

$$\frac{|f|}{\log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right)\beta} \geq \frac{\log_{1+\epsilon}(1/\delta m)}{\log_{1+\epsilon}\left(\frac{1+\epsilon}{\delta}\right)}.$$

By setting $\delta = (1+\epsilon)^{1-1/\epsilon} m^{-1/\epsilon}$, this quantity becomes $1-\epsilon$, as desired. □

We conclude by emphasizing that Algorithm 1 does not rely on any specific properties of network flows. In particular, there is no residual or admissible network. This illustrates a general strategy for solving linear programs that resemble (P) and (D) (known as packing/covering LPs). A similar technique can be used to solve the maximum concurrent flow problem, in which we seek to maximize the minimum amount of flow between any source-sink pair.

3 Summary

In this lecture, we looked at the maximum multicommodity flow problem and an algorithm developed by Garg and Könemann [GK07]. This is an approximation algorithm, and both the algorithm and its analysis rely on linear program duality.

References

[GK07] Naveen Garg and Jochen Köenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.