The Congestion Problem of the Single-Source Unsplittable Flow

Yuchong Pan

June 16, 2020

Abstract

In this paper, we survey an algorithm given by [1] that induces an upper bound 2 of congestion for the single-source unsplittable flow problem, assuming the *cut condition* and the *no bottleneck assumption*. The essence of the algorithm is to transform *any* feasible flow satisfying all demands into an *unsplittable* flow, by augmenting flow along *alternating cycles* and moving terminals, while increasing each edge capacity by at most the maximum demand.

1 Problem and Assumptions

The congestion problem of the single-source unsplittable flow can be formulated by the following. Let $G = (V, E)$ be a directed graph. Let $c : E \to \mathbb{R}^+$ be the edge capacities. Let $s \in V$ be the source. Let (t_i, d_i) for $i \in [k]$ be k commodities, for some $k \in \mathbb{N}$, each with terminal $t_i \in V$ and demand $d_i \in \mathbb{R}^+$. We note that a vertex may contain multiple terminals. The goal of the single-source unsplittable flow problem is to route d_i units of commodity i from s to t_i *along a single path*, for each $i \in [k]$. The *congestion* question asks the following:

Question 1. *What is the smallest $\alpha \geq 1$ such that if we multiply all edge capacities by α, an unsplittable flow satisfying all demands exists?*

Our explication of the problem and the algorithm follows the terminology used in [1], which we define in the following:

Definition 1. A *flow* is a function $f : E \to \mathbb{R}^+ \cup \{0\}$ such that the *net inflow* at any $v \in V \setminus \{s\}$ is nonnegative and at most the sum of the demands at v, i.e.,

$$0 \leq \sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e) \leq \sum_{\substack{i \in [k] \\ t_i = v}} d_i, \qquad \forall v \in V \setminus \{s\}.$$

Definition 2. We say that a flow $f : E \to \mathbb{R}^+ \cup \{0\}$ is *feasible* if $f(e) \leq c(e)$ for all $e \in E$.

Definition 3. We say that a flow $f : E \to \mathbb{R}^+ \cup \{0\}$ satisfies a subset $I \subseteq [k]$ of commodities if the *net inflow* at any $v \in V$ equals the sum of the demands at v that belong to I, i.e.,

$$\sum_{e \in \delta^-(v)} f(e) - \sum_{e \in \delta^+(v)} f(e) = \sum_{\substack{i \in I \\ t_i = v}} d_i, \qquad \forall v \in V.$$

Definition 4. We say that a flow $f : E \to \mathbb{R}^+ \cup \{0\}$ is *unsplittable* if each commodity $i \in [k]$ is routed along a single path from s to t_i.

We assume the *cut condition*, which states the following:

For any $S \subseteq V \setminus \{s\}$, the total demand of terminals within S is at most the total capacity of the edges entering S, i.e.,

$$\sum_{\substack{i \in [k] \\ t_i \in S}} d_i \leq \sum_{e \in \delta^-(S)} c(e), \qquad \forall S \subseteq V \setminus \{s\}.$$

A fundamental theorem in the network flow theory shows that there exists a flow if and only if the *cut condition* is satisfied. Let d_{max} be the maximum demand over all commodities $i \in [k]$. Let c_{min} be the minimum edge capacities over all edges $e \in E$. Assuming the cut condition, the algorithm to be presented transforms *any* feasible flow satisfying all demands to an *unsplittable* flow while increasing each edge capacity by at most d_{max}. Hence, if we further assume the *no bottleneck assumption*, i.e. $d_{max} \leq c_{min}$, a feasible flow satisfying all demands is guaranteed to exist, implying that the congestion upper bounded by 2.

2 Algorithm

We begin the explication of the algorithm with the following important definitions:

Definition 5. We say that an edge $e = (u, v) \in E$ is *singular* if v and all vertices reachable from v have out-degree at most 1, i.e. the vertices reachable from v form a directed path.

Definition 6. We say that a terminal t_i for some $i \in [k]$ is *regular* if $d_i > f(e)$ for all $e \in \delta^-(v)$. Otherwise, we say that t_i is *irregular*.

Let $f : E \to \mathbb{R}^+ \cup \{0\}$ be a feasible flow that satisfies all demands. We summarize the algorithm by the following steps:

1. While there exists a cycle C in the digraph G such that $f(e) > 0$ for all $e \in E(C)$, we eliminate C by decreasing $f(e)$ by $\min_{e \in E(C)} f(e)$ for each $e \in E(C)$.

   We note that modifying the flow f by this step does not change the net inflow at any $v \in V(C)$. For any $v \in V(C)$ has an incoming edge and an outgoing edge in $E(C)$, the flow on each of which is decreased by the same amount. Hence the updated flow f remains feasible and still satisfies all demands. We assume that f is *acyclic* from now on, in the sense that there does not exist a cycle C in G such that $f(e) > 0$ for all $e \in E(C)$.

   We remove all edges $e \in E$ with $f(e) = 0$. In addition, in the following explication, we remove e from E whenever $f(e)$ vanishes. Hence we assume that G is acyclic.

2. While there exist $i \in [k]$ and $e = (u, t_i) \in \delta^-(t_i)$ such that $f(e) \geq d_i$, we move t_i to u and decrease $f(e)$ by d_i. For each $i \in [k]$, if $t_i = s$ after the move, then we remove commodity i from the set of commodities.

   Let t_i' denote the original t_i. For each iteration, the net inflow at t_i' is decreased by d_i, and the net inflow at u is increased by d_i. Since f satisfies demand i before the iteration, then the original net inflow at t_i' equals the sum of the demands at t_i', including d_i. Since we move t_i to u, then the sum of the demands at t_i' is decreased by d_i, and that at u is increased by d_i. Hence f remains feasible and still satisfies all demands. We assume that terminal t_i is regular from now on. This implies the following important claim, to be used by the next step:

   Claim 1. *At the end of step 2, for all $v \in V$, if v contains a terminal, then v has at least two incoming edges.*

Proof. Let $v \in V$ be such that v contains a terminal, say $i \in [k]$. Since f satisfies all demands, then the net inflow at v equals the sum of the demands at v, and is hence at least d_i. Since $f(e) < d_i$ for each $i \in [k]$ and $e \in \delta^-(t_i)$, then there exist at least two incoming edges. □

After the description of step 3, we will prove the following claim:

Claim 2. *At the end of each iteration in step 3, f satisfies all demands. Furthermore, for all $v \in V$, if v contains an irregular terminal, then v also contains a regular terminal.*

This claim directly implies the following claim, to be used by the next step. For the existence of an irregular terminal entails the existence of a regular terminal by Claim 2.

Claim 3. *At the end of each iteration, for all $v \in V$, if v contains a terminal, then v has at least two incoming edges.*

3. We repeat the following sub-steps until all terminals reach the source s:

   (a) *Find an alternating cycle.* Let $v \in V$ be arbitrary. We follow outgoing edges as long as possible from v until we reach a vertex with out-degree 0. Since G is acyclic, then this process terminates. We call such a path a *forward path*. Since the flow f satisfies the demands, then a vertex v contains a terminal $v \in V$ has no outgoing edges. Hence the above process terminates at some v that contains a terminal. By Claim 1 and Claim 3, v has at least two incoming edges. Let e be an incoming edge of v that does not occur in the proceeding forward path.

   We follow *singular* incoming edges as long as possible from e. Since G is acyclic, then this process terminates. We call such a path a *backward path*. Let $e' = (v', u) \in V$ be the vertex at which the above process stops. We show the following important claim:

   Claim 4. *v' has at least two outgoing edges.*

   Proof. We have the following two cases:

   i. $v' = s$. Suppose for the sake of contradiction that e' is the only outgoing edge of v'. Since e' is singular, then the vertices reachable from u form a directed path. Since e' is the only outgoing edge of s, then G is a directed path. This contradicts the previous claim that v has at least two incoming edges.

   ii. Since E consists of edges e with $f(e) > 0$ only, then there exists an sv'-path P such that $f(e) > 0$ for all $e \in E(P)$. Hence e' has an incoming edge, namely $\tilde{e} = (u', v')$. By the maximality of the backward path, $\tilde{e}$ is not singular. Since edges along the backward path are all singular, then v' has at least two outgoing edges.

   This completes the proof. □

   Let e'' be an outgoing edge of v' that does not occur in the proceeding backward path. We repeat the above procedure from e'' to construct forward and backward paths alternately, until we encounter a vertex w that already belongs to a previous forward or backward path. This process terminates by the pegionhole principle. Hence, the constructed forward and backward paths form a cycle (in the underlying undirected graph). If the incoming and outgoing paths of w on the constructed alternating cycle have the same direction, we combine the two paths into one. We call such a cycle an *alternating cycle*.

(b) *Augment flow along the alternating cycle.* Let C be the alternating cycle found in the previous sub-step. Let

$$\varepsilon_1 = \min\{f(e) : e \in E(C), e \text{ is on a forward path}\},$$
$$\varepsilon_2 = \min\{d_i - f(e) : i \in [k], e = (u, t_i) \in E(C), e \text{ is on a backward path}\}.$$

Let $\varepsilon = \min(\varepsilon_1, \varepsilon_2)$. Since $f(e) > 0$ for all $e \in E(C)$, then $\varepsilon_1 > 0$. We have $\varepsilon_2 > 0$ by definition. Hence $\varepsilon > 0$. We decrease the flow along the forward paths on C, and increase the flow along the backward paths of C, both by ε.

(c) *Move terminals.* We move a terminal t_i to u along $e = (u, t_i)$ and decrease $f(e)$ by d_i if at least one of the following two conditions is true, with preference to (i):

i. e is singular and $f(e) = d_i$;

ii. e is not singlular and $f(e) \geq d_i$.

In either case, we decrease $f(e)$ by d_i.

We now prove Claim 2, which is crucial to the validity of step 3.

Proof of Claim 2. Firstly, it follows from the same argument as in step 2 that the flow f satisfies all demands at the end of each iteration.

Let $v \in V$ be such that v contains an irregular terminal, say $i \in [k]$. Then there exists $e = (u, v) \in \delta^-(v)$ with $f(e) > d_i$. For if $f(e) = d_i$, then terminal i should have been moved to u in the previous iteration by the rules of moving terminals. By the rules of augmenting flow, $f(e)$ cannot be augmented from below d_i to above d_i. Hence, terminal i was moved along an outgoing edge of v in a previous iteration. Suppose that terminal i was moved to v along $(v, w) \in \delta^+(v)$ during iteration j.

Since $f(e) > d_i$ during the *current* iteration and since $f(e)$ cannot be augmented from below d_i to above d_i by the rules of augmenting flow, then $f(e) > d_i$ at the end of iteration j. Since terminal i was not moved to u along (u, v) during iteration j, then e is singular. For $f(e) > d_i$ and e being non-singular imply that t_j would be moved to u during iteration i by the rules of moving terminals. By the definition of singular edges, (v, w) is also singular, and (v, w) is the only outgoing edge of v.

Since terminal i is moved to u along e during the *current* iteration, then (v, w) vanishes after moving i. Since (v, w) is the only outgoing edge of v, then the out-degree of v at the end of the *current* iteration becomes 0. Since $f(e)$ satisfies all demands at v and since $f(e) > d_i$, then there exists a terminal $i' \neq i$ contained in v. Note that after moving the first irregular terminal to v, the out-degree of v becomes 0. Since we never add edges, then the out-degree of v remains 0 after that. Hence, there exists at most one irregular terminal at v. Hence, terminal i' is regular. This completes the proof. □

4. For each $i \in [k]$, we define the path to route commodity i to be the reverse path to the one given by moving t_i in steps 2 and 3. This completes the algorithm.

3 Correctness

The validity of the algorithm has been shown as we present the algorithm in Section 2; that is, the algorithm successfully finds an alternating cycle in each iteration of step 3. It remains to prove the following theorem:

Theorem 1. *The algorithm presented in Section 2 finds an unsplittable flow for each commodity $i \in [k]$. Furthermore, the total flow on any edge e exceeds the initial flow on e (hence the capacity of e) by at most the maximum demand d_{max}.*

Proof. Since the flow for each commodity $i \in [k]$ is entirely along the reverse path to the one given by moving t_i to s, then the flow from s to t_i is unsplittable.

By the rules of augmenting flow and of moving terminals, the flow on an edge $e \in E$ increases only if e is on a backward path of an alternating cycle, and hence only if e is singular. This implies that the total flow on an edge $e \in E$ *before e becomes singular* does not exceed the initial flow of e and hence the capacity of e. By the proof of Claim 2, at most one commodity is ever moved along a singular edge. Since we never add edges, a singular edge cannot become non-singular at any stage of the algorithm. This implies that the total flow on any edge $e \in E$ is at most the capacity of e plus the maximum demand. This completes the proof. □

4 Tightness

The tightness of Theorem 1 is witnessed by the following set of instances of the single-source unsplittable congestion problem: For each $q \in \mathbb{N}$, we construct an instance with $d_{max} = 1$ such that any solution of unsplittable flows violates an edge by at most $1 - \frac{1}{q}$.

Let $q \in \mathbb{N}$. Let $V = \{0, \ldots, q+1\}$ and $E = \{(0,i) : i \in [q]\} \cup \{(i, q+1) : i \in [q]\}$. Let $s = 0$. Let $c(e) = 1$ for all $e \in E$. Let $(t_i, d_i) = (i, 1 - \frac{1}{q})$ for all $i \in [q]$ and $(t_{q+1}, d_{q+1}) = (q+1, 1)$ be commodities. Any flow from s to terminal t_i for $i \in [q]$ is unique and unsplittable, i.e. along the edge (s, i). Furthermore, any unsplittable from s to terminal t_{q+1} uses one of the edges $(0, i)$ amongst $i \in [q]$, say $(0, j)$. Therefore, the total flow on $(0, j)$ equals $1 + 1 - \frac{1}{q}$, whereas the capacity of $(0, j)$ equals 1. This shows that there exists an edge on which the total flow exceeds the capacity by $1 - \frac{1}{q}$. This completes the instance.

5 Running Time

Let $n = |V|$ and $m = |E|$. Since the augmentation in each iteration of step 3 of the algorithm removes at least one edge (either immediately or after moving terminals), then the number of iterations in step 3 is upper bounded by m. Finding an alternating cycle in each iteration takes $O(n)$ time. Since each terminal moves at most n steps to s, then the running time for moving terminals is $O(kn)$. Computing ε_2 during each iteration takes $O(k)$ time. Therefore, the running time for the entire algorithm is $O(nm + km)$.

For computing ε_2, we may maintain a binary heap and update $\min\{d_i - f(e) : e \in \delta^-(t_i)\}$ each time terminal t_i is moved with running time $O(k \log k)$. Since each terminal is moved at most n times, then the total update time to the binary heap is $O(kn \log k)$. Hence, the running time for the entire algorithm is $O(nm + kn \log k)$.

References

[1] Y. Dinitz, N. Garg, and M. X. Goemans, *On the single-source unsplittable flow problem*, Combinatorica, 19 (1999), pp. 17–41.