CO 351 Review Questions (Part II)

David Duan

2019 Fall

Contents

1.4.1

Prove flow decomposition: Given $D = (N, A)$, s,t, and integer capacities, if x is an s,t-flow of value k and x is integral, then x is the sum of characteristic vectors of k s,t-dipaths and any number of dicycles.

1.4.2

Prove Menger's Theorem (Arc-Disjoint Version): Given $D = (N,A)$, nodes s,t, the maximum number of arc-disjoint s,t-dipaths is equal to the minimum number of arcs that disconnects s from t.

1.4.3

Prove Menger's Theorem (Node-Disjoint Version): If st is not an arc, then the maximum numbers of node-disjoint s,t-dipaths is equal to the minimum number of node whose removal disconnect s from t. (Hint: Apply transformation)

1.4.4 Prove Konig's Theorem: In a bipartite graph, $\nu(G) = \tau(G)$.

1.5 Real-Life Applications

1.5.1 Describe how we could solve the matrix rounding problem using max flow.

1.5.2 Describe how we could solve the maximum closure problem using max flow.

1.6 Preflow-Push Algorithm

1.6.1 What is the drawback of FF? What is the intuition behind PFP?

1.6.2 What is the definition for s,t-preflow and **excess** at $v \in N$?

1.6.3 Given $D = (N,A)$, capacities c, flow x, describe the residual digraph.

1.6.4 What does it mean for a set of heights to be compatible with a preflow x?

1.6.5 Describe the preflow-push algorithm.

1.7 Correctness of Preflow-Push

1.7.1 Prove: If preflow x and height h are compatible, then D' has no s,t-dipath.

1.7.2 Prove: If x is a feasible flow with compatible heights h, then x is a max flow.

1.7.3 Prove: The algorithm maintains a preflow and a height function that are compatible with each other.

1.8 Termination of Preflow-Push

1.8.1 Prove: If u has excess, i.e., $e(u) > 0$, then there is a u,s-dipath in D'.

1.8.2 Prove: Throughout the algorithm, $h(u) \leq 2|N| - 1$ for all $u \in N$.

1.8.3 Prove: The total number of relabel operations is at most $2|N| \times |N| = 2|N|^2$.

1.8.4 Define: Saturating push, non-saturating push.

1.8.5 Prove: The number of saturating pushes throughout the algorithm is at most $2|N||A|$.

1.8.6 Prove: The number of non-saturating pushes throughout the algorithm is $\leq 4|N|^2|A|$.

1.8.7 Prove: PFP terminates in $2|N|^2+2|N|^2|A| + 4|N|^2|A| = 8|N|^2|A| \approx 8|N|^4$ operations.

2 Global Minimum Cut

2.1 Global Minimum Cut

2.1.1 Describe the global min cut problem. What's the motivation for an efficient algorithm?

2.1.2 Define: X,t-cut, s-cut.

2.1.3 Describe the generic algorithm for minimum s-cut.

2.1.4 Prove: The generic algorithm above solves the minimum s-cut problem.

2.2 Hao-Orlin

2.2.1 Define: X-preflow.

2.2.2 What does it mean for height h to be compatible with an X-preflow?

2.2.3 Define: level, cut level.

2.2.4

Prove: If $\delta(S)$ is an X,t-cut with $\delta_{D'}(S) = \varnothing$ and $e(v) = 0$ for all $v \in N\setminus (S \cup \{t\})$ then $\delta(S)$ is a minimum $X,t $-cut.

2.2.5

Prove: If ℓ is a cut level and $e(v) =0$ for all v with $h(v) < \ell$, except t, then $\{v:h(v) \geq \ell\}$ is a min X,t-cut.

2.2.6 Describe the Hao-Orlin algorithm.

2.3 Correctness of Hao-Orlin

2.3.1 Prove: The non-empty levels less than $|N|$ are consecutive.

2.3.2 Prove: The X-preflow and height h are always compatible.

2.3.3 Prove: $h(v) \leq |N| -2$ for all $v \notin X$.

2.3.4 Prove: ℓ is always a cut level.

2.3.5 The stored cuts in each iteration are minimum X,t-cuts.

2.4 Global Min Cut in Undirected Graphs (Karger's Algorithm)

2.4.1 Describe Karger's algorithm. What's the intuition?

2.4.2

Prove: Let $\delta(S^*)$ be a global min cut. The probability that the algorithm produces $\delta(S^*)$ is $\geq \frac{1}{\binom{|V|}{2}}$.

2.4.3

Prove: The probability that that algorithm produces $\delta(S^*)$ after $k|V|^2$ runs is at least $1-e^{-2k}$ where $k \geq 1$.

1 Maximum Flow

1.1 Maximum Flow Problem

1.1.1 Describe the maximum flow problem?

Given a digraph $D = (N, A)$, constraints $c \in \mathbb{R}^A$, nodes $s, t \in N$ as source and sink, we wish to maximize the total flow from s to t.

1.1.2 Give the primal and dual LP of the maximum flow problem.

The primal LP is given by

$$\begin{array}{lll} \max & x(\delta(s)) - x(\delta(\bar{s})) & \\ s.t. & x(\delta(\bar{v})) - x(\delta(v)) = 0 & \forall v \in N \setminus \{s, t\} \\ & 0 \leq x_e \leq c_e & \forall e \in A \end{array}$$

The dual LP is equivalent to

$$\begin{array}{lll} \min & c^T z & \\ s.t. & y_s = 1, y_t = 0 & \\ & -y_u + y_v + z_{uv} \geq 0 & \forall uv \in A \\ & z_e \geq 0 & \forall e \in A \end{array}$$

where y is the dual variable for the flow constraints and z is the for capacities constraints.

1.1.3 Define: residual capacity, augmenting path.

The *residual capacity* $\gamma(P)$ of the s, t-dipath P is

$$\min(\{c_{uv} - x_{uv} : uv \text{ is a forward arc of } P\} \cup \{x_{uv} : uv \text{ is a backward of } P\}.$$

An augmenting path is an s, t-dipath where $\gamma(P) > 0$.

1.1.4 Given $D = (N, A)$, capacities c, and flow x, define the *residual graph* D'.

$N' = N$, for each arc $uv \in A$,

- If $x_{uv} < c_{uv}$, add arc uv to A' with residual $c_{uv} - x_{uv}$.
- If $x_{uv} > 0$, add arc vu to A' with residual x_{uv}.

1.2 Ford-Fulkerson and Max-Flow Min-Cut

Note: FF runs in $O(|E| \cdot f^*)$ where f^* is the max flow value.

1.2.1 Describe Ford-Fulkerson algorithm.

0. Let $x_e = 0$ for all $e \in A$.

1. Construct the corresponding residual graph D'.
2. If the residual graph D' contains an s,t-dipath P, push $\gamma(P)$ flow along P. Go back to (1).
3. Let S be the set of vertices reachable from s in D'. STOP. x is the max flow and $\delta(S)$ is the min cut.

1.2.2 Under what condition will FF terminate?

First, if c is integral, then at each step we increase the flow by at least 1. By induction, it terminates. If c is rational, we could multiply everything by the GCD of the denominators and make c integral, so it will terminate as well. However, if c is irrational, then it is possible that it does not terminate.

We say FF runs in pseudo-polynomial time, that is, it is polynomial in the magnitude of input but exponential in the size of the input.

1.2.3 Describe Edmonds-Karp algorithm. What's the key difference between EK and FF?

The algorithm is identical to FF, except we always pick the augmenting path with the fewest number of arcs. This guarantees the termination of the algorithm.

1.2.4 Given $D = (N, A)$, nodes s and t, capacities c, show the value of any s,t-flow is at most the capacity of any s,t-cut.

For any s,t-flow x and any s,t-cut $\delta(S)$, the net flow of S is equal to the net flow of s as the net flows on other nodes are 0 by the problem setting. Observe

$$x(\delta(s)) - x(\delta(\bar{s})) = x(\delta(S)) - x(\delta(\bar{S})) \le x(\delta(S)) \le c(\delta(S)).$$

Thus, the value of any s,t-flow is at most the capacity of $\delta(S)$. □

1.2.5 Use (6) to prove MFMC: Given $D = (N, A)$, nodes s and t, capacities c, show the maximum value of any s,t-flow is equal to the minimum capacity of an s,t-cut.

Let x be a max s,t-flow. Then there is no s,t-dipath in the residual digraph D', so there is an empty s,t-cut $\delta_{D'}(S)$ in D'.

- If $uv \in \delta_D(S)$, uv is not an arc in D', so $x_{uv} = c_{uv}$.
- If $uv \in \delta_D(\bar{S})$, vu is not an arc in D', so $x_{uv} = 0$.

Therefore, the value of x is $x(\delta(S)) - x(\delta(\bar{S})) = c(\delta(S))$. By the previous proposition, x is a max flow and $\delta(S)$ is a min cut. □

1.2.6 As a corollary of MFMC, show that FF gives a s,t-flow when it terminates.

Since FF terminated, there is no augmenting path, so our flow has used up the capacity of some cut $\delta(S)$ in D. By MFMC, the capacity of the min cut equals the max flow, the result follows. □

1.3 Maximum Flow with Lower Bounds

Suppose we introduce non-negative lower bounds $\ell_e \in \mathbb{R}^A$ for the arcs. That is, a flow must satisfy $\ell_e \leq x_e \leq c_e$ for all $e \in A$.

1.3.1 Modify FF to solve the maximum flow with lower bound problem.

Given a feasible flow, we form the residual digraph D', where for backward arcs, we put residual $x_e - \ell_e$ instead of x_e because we want to limit how much flow we could reduce. (You could interpret the no lower bound version as having a lower bound of zero.) The rest of FF is exactly the same.

1.3.2 Similar to (6): Show the value of any s, t-flow is at most $c(\delta(S)) - \ell(\delta(\bar{S}))$.

For any s, t-flow x and any s, t-cut $\delta(S)$, we have $x(\delta(\bar{S})) \geq \ell(\delta(\bar{S}))$, so

$$x(\delta(s)) - x(\delta(\bar{s})) = x(\delta(S)) - x(\delta(\bar{S})) \leq x(\delta(S)) - \ell(\delta(\bar{S})) \leq c(\delta(S)) - \ell(\delta(\bar{S})). \quad \square$$

1.3.3 Similar to (7): State the generalized MFMC using the above result.

Given $D = (N, A)$, c, ℓ, and nodes s, t, there is a maximum s, t-flow x whose value is the same as the minimum value of $c(\delta(S)) - \ell(\delta(\bar{S}))$ provided there is a feasible solution.

1.3.4 Prove D is infeasible iff there is an s, t-cut $\delta(S)$ s.t. $c(\delta(S)) - \ell(\delta(\bar{S})) < 0$.

If $c(\delta(S)) - \ell(\delta(\bar{S})) < 0$, by MFMC, the max flow is negative, a contradiction. Now suppose the network is feasible. Then we cannot have an s, t-cut with $c(\delta(S)) - \ell(\delta(\bar{S})) < 0$ because that sets an upper bound for the flow, again a contradiction. $\square$

1.4 Combinatorial Applications

1.4.1 Prove flow decomposition: Given $D = (N, A)$, s, t, and integer capacities, if x is an s, t-flow of value k and x is integral, then x is the sum of characteristic vectors of k s, t-dipaths and any number of dicycles.

If $k = 0$, i.e., x is a circulation, then x is the sum of characteristic vectors of dicycles. If $k > 0$, then there is an s, t-dipath P, using our proof for shortest dipath; $x - x^P$ is an s, t-flow of value $k - 1$. We are done by induction on k. $\square$

1.4.2 Prove Menger's Theorem (Arc-Disjoint Version): Given $D = (N, A)$, nodes s, t , the maximum number of arc-disjoint s, t-dipaths is equal to the minimum number of arcs that disconnects s from t.

If there are k arc-disjoint s, t-dipaths, then we must remove at least one arc from each dipath. The inequality $\max \leq \min$ is thus trivial. We now show the equality.

Apply MFMC with capacity $c = \mathbf{1}$. By MFMC, there is an s,t-flow with the same value as the capacity of an s,t-cut $\delta(S)$, say k. We may assume that x is integral as c is integral. By flow decomposition, x is the sum of k s,t-dipaths and some dicycles. Since $c = 1$, each arc is used on at most 1 s,t-dipath. Hence, these s,t-dipaths are all arc-disjoint. If we remove all k arcs in $\delta(S)$, then S is disconnected from t. □

1.4.3 Prove Menger's Theorem (Node-Disjoint Version): If st is not an arc, then the maximum numbers of node-disjoint s,t-dipaths is equal to the minimum number of node whose removal disconnect s from t. (Hint: Apply transformation)

We transform the node-disjoint version into arc-disjoint version. For every node v, generate two nodes v^+ and $v-$ with a single arc from v^+ to $v-$. Arcs of the form uv maps to u^+v- with 1 and all other arcs have capacity ∞. Then we can simply apply Menger's Theorem for arc-disjoint version and conclude the proof: min-cut cannot use arcs with ∞ and thus only newly created edges, which corresponds to an s,t-separator. □

1.4.4 Prove Konig's Theorem: In a bipartite graph, $\nu(G) = \tau(G)$.

For any graph G, $\nu(G) \leq \tau(G)$ holds trivially. Now, let A, B be a bipartition of V and create a, b with arcs aA, Bb. Each newly created edge has capacity 1. We then orient edges from a to b; original arcs have capacity ∞.

A maximum matching consists of a maximum number of internally disjoint a,b-paths. By Menger's Theorem, this is precisely the set of some minimum separator S. Notice that S is actually a vertex cover by definition, so the result follows. □

1.5 Real-Life Applications

1.5.1 Describe how we could solve the matrix rounding problem using max flow.

Suppose we want to round the matrix $M \in \mathbb{R}^{m,n}$ elements either by taking the ceiling/floor so that the sum of the columns α_i, $1 \leq i \leq m$, and sum of rows β_j, $1 \leq j \leq n$, are floored/ceilinged.

To transform this into a max flow problem:

1. Let s, t be dummy nodes and create nodes r_i, c_j for $1 \leq i \leq m$, $1 \leq j \leq n$.
2. Add an arc from r_i to c_j with $\ell_{r_i c_j} = \lfloor a_{ij} \rfloor$ and $c_{r_i c_j} = \lceil a_{ij} \rceil$.
3. All arcs sr_i have lower bound 0 and capacity floor/ceiling α_i
4. All arcs $c_j t$ have lower bound 0 and capacity floor/ceiling β_j

We then look for a feasible flow saturating all arcs incident with s and t.

1.5.2 Describe how we could solve the maximum closure problem using max flow.

A closure is a set of nodes such that $\delta(S) = \varnothing$. Suppose we are given a set of tasks, each with some benefit (positive or negative). However, to perform some task, you need to also perform other tasks dependent on it. Form a digraph $D = (N, A)$ with node weights $w \in \mathbb{R}^N$; an arc uv represents "if we take u we must also take v". Our goal is to find a closure with max total weight.

To transform this into a max flow problem:

1. Add s, t be dummy nodes.
2. Add arcs su with capacity $w(u)$ if $w(u) > 0$.
3. Add arcs vt with capacity $-w(v)$ if $w(v) < 0$.
4. Original arcs have capacity ∞.

We have the following observations:

1. An s,t-cut $\delta_{D'}(S)$ in D' has finite capacity iff $S \setminus \{s\}$ is a closure in D. (Because there exists an arc in $\delta(S)$ with infinite capacity iff such an arc existed in the original graph.)
2. The weight of the closure is $c(\delta(s)) - c(\delta(S))$. (There is an arc from s to every u with $b_u > 0$ and an arc from every v with $b_v < 0$ to t. Nodes not in the closure are included in both are thus get cancelled.)
3. $c(\delta(s))$ is a constant and $c(\delta(S))$ is the capacity of an s,t-cut.

Thus, to maximize the weight of closure $c(\delta(s)) - c(\delta(S))$, we minimize $c(\delta(S))$, i.e., find the min cut. We can solve this using max flow.

1.6 Preflow-Push Algorithm

1.6.1 What is the drawback of FF? What is the intuition behind PFP?

FF/EK does global adjustments, which could be inefficient sometimes. PFP, on the other hand, makes local adjustments.

1.6.2 What is the definition for s,t-preflow and excess at $v \in N$?

An s,t-preflow is a flow that satisfies capacity constraints and the in-flow is greater than or equal to the out-flow for each intermediate nodes. If in-flow is greater than out-flow, we call the extra amount excess.

Intuitively, an s,t-preflow is a flow that "makes sense", i.e., does not violate constraints or having more out-flow than in-flow.

1.6.3 Given $D = (N, A)$, capacities c, flow x, describe the residual digraph.

The residual graph is identical to the one for FF.

1.6.4 What does it mean for a set of heights to be compatible with a preflow x?

$h : N \to \mathbb{N}_0$. Height h is compatible with a preflow x if

1. $h(s) = |N|$, $h(t) = 0$, and
2. $h(v) \geq h(u) - 1$ for $uv \in A(D')$.

1.6.5 Describe the preflow-push algorithm.

0. Initialization.
    a. Set $h(s) = |N|$ and $h(v) = 0$ for every $v \in N \setminus \{s\}$.
    b. Set preflow x with $x_e = c_e$ for every $e \in \delta_D(s)$ and $x_e = 0$ otherwise.
1. While there exists $u \in N \setminus \{s, t\}$ with excess, i.e., $e_u > 0$,
    a. If there exists $uv \in A(D')$ where $h(v) = h(u) - 1$, push $\min\{r_{uv}, e_u\}$ on uv.
    b. Otherwise, increment $h(u)$ by 1. (Relabel operation.)

1.7 Correctness of Preflow-Push

1.7.1 Prove: If preflow x and height h are compatible, then D' has no s,t-dipath.

Suppose not, so an s,t-dipath exists in D'. Let $s = v_0, v_1, \ldots, v_k = t$ in D' and $h(v_{i+1}) \geq h(v_i) - 1$ (compatible). Adding all these inequalities gives $h(v_k) \geq h(v_0) - k$, so $0 \geq |N| - k$ and thus $|N| \leq k$, a contradiction as there are $k + 1$ nodes in the path but only $|N|$ nodes in the digraph. $\square$

1.7.2 Prove: If x is a feasible flow with compatible heights h, then x is a max flow.

Since x is compatible with h, by the lemma above, D' has no s,t-dipath. Since there is no augmenting path in D (so no extra flow is available) and x is feasible, x is indeed optimal. $\square$

1.7.3 Prove: The algorithm maintains a preflow and a height function that are compatible with each other.

At initialization, we have a preflow x where $x_e = c_e$ for every $e \in \delta(s)$ and 0 otherwise.

- By construction, $h(s) = |N|$ and $h(t) = 0$.
- Also, $x_{sv} = c_{sv}$ for arcs in D, so only vs is in D'. Observe $h(s) = |N| \geq -1 = h(v) - 1$.
- For arcs x_{uv} where $u \neq s$, $x_{uv} = 0$, so $uv \in D'$. Now $h(v) = 0 \geq -1 = h(u) - 1$.

Thus, the preflow and height are compatible at initialization. Now suppose there exists $u \in N \setminus \{s, t\}$ with excess, i.e., $e_u > 0$.

Consider a push operation, i.e., when there exists $uv \in A(D')$ satisfying $h(v) = h(u) - 1$, we push flow $\min\{e_u, r_{uv}\}$ along uv. Since we are pushing the minimum between excess and residual, it is always true that $x(\delta(\bar{u})) \geq x(\delta(u))$ and $x_{uv} \leq c_{uv}$; exactly one of these becomes an equality.

- If we push residual, then $x_{uv} = c_{uv}$. The only possible new arc in D' is vu but $h(u) > h(v)$ so we are fine.
- If we push excess, then $e_u = 0$. Then $x_{uv} > 0$, so the only possible new arc is also vu and as above we are fine.

Thus the preflow and height are compatible after a push operation.

Consider a relabel operation, i.e., when every $uv \in A(D')$ satisfies $h(v) > h(u) - 1$, we increment $h(u)$ by 1. Then before relabeling, it must be that every $uv \in A(D')$ satisfies $h(v) \geq h(u)$. Then adding 1 to $h(u)$, we have $h(v) \geq h(u) - 1$ for all arcs $uv \in A(D')$. Thus the preflow and height are compatible after a relabel operation.

Hence, if the algorithm terminates, this guarantees the correctness of the algorithm. $\square$

1.8 Termination of Preflow-Push

1.8.1 Prove: If u has excess, i.e., $e(u) > 0$, then there is a u, s-dipath in D'.

We show the contrapositive. Let T be the set of all nodes $v \in N$ with no v, s-dipath in D' and we show that $e(v) = 0$. Suppose $\delta_{D'}(T) \neq \varnothing$, i.e., there exists vz where $v \in T$ and $z \in N \setminus T$. But then v could reach s via z. Thus $\delta_{D'}(T) = \varnothing$, so every in-arc is empty and every out-arc is full for T. It follows that $\sum_{v \in T} e(v) = x(\delta(\bar{T})) - x(\delta(T)) = 0 - c(\delta(T)) \leq 0$. $\square$

1.8.2 Prove: Throughout the algorithm, $h(u) \leq 2|N| - 1$ for all $u \in N$.

Suppose that at some point in the algorithm the height $h(u)$ increases to $2|N|$. Since a relabel operation only occurs when there is excess, this implies that $e(u) > 0$. By the previous lemma, there is a u, s-dipath in D', which has length at most $|N| - 1$. By compatibility, height decreases by at most 1 in each arc resulting in $h(s) > |N|$, a contradiction. $\square$

1.8.3 Prove: The total number of relabel operations is at most $2|N| \times |N| = 2|N|^2$.

There are $|N|$ nodes in D, each can have at most $2|N|$ relabel operations (from 0 to $2|N| - 1$), so the overall upper bound is $2|N|^2$. $\square$

1.8.4 Define: Saturating push, non-saturating push.

If we push r_{uv} on arc $uv \in A(D')$ then it is a saturating push (the arc is saturated as we use up its capacity); if we push $e(u)$ then it is called a non-saturating push.

1.8.5 Prove: The number of saturating pushes throughout the algorithm is at most $2|N||A|$.

Suppose we have a saturating push on uv, then before the push, $h(u) = h(v) + 1$. Observe uv disappears from D' as $x_{uv} = c_{uv}$. In order to push on uv again, we need to first push flow on vu. To do so, we need to relabel v at least twice to obtain $h(v) = h(u) + 1$. Thus, between two saturated pushes on the same edge, at least two relabel operations need to occur. By previous lemma, the maximum number of relabel operations on node v cannot exceed $2|N|$, so up to $|N|$ saturating pushes are possible on uv. There are $2|A|$ arcs in D' counting both forward and backward arcs, so the number of saturating pushes is $2|N||A|$. □

1.8.6 Prove: The number of non-saturating pushes throughout the algorithm is $\leq 4|N|^2|A|$.

Define the function $\Phi(x, h)$ which counts the total height for nodes with excess, i.e.,

$$\Phi(x, h) = \sum_{v \in N, e(v) > 0} h(v).$$

At initialization, $\Phi = 0$, because all nodes with excess have height zero. Note the function never becomes negative as we are summing up non-negative heights. We want to determine the effect on $\Phi(x, h)$ for each type of operation.

Each relabel operation is done on a node with excess, so $\Phi(x, h)$ goes up by one. There are $2|N|^2$ relabel operations by previous lemma, so the maximum increases for relabeling is $2|N|^2$ throughout the algorithm.

Each saturating push (i.e., push r_{uv}) on uv decreases $e(u)$ and increases $e(v)$. If $e(v) = 0$ before the push, we add $h(v)$ to $\Phi(x, h)$ after the push. By previous lemma, $h(v) \leq 2|N| - 1$, so we add at most $2|N| - 1$ over the algorithm. By another lemma, there is no more than $2|N||A|$ saturating pushes, so the maximum increases from saturating pushes is $2|N||A|(2|N| - 1) \leq 4|N|^2|A|$.

Each non-saturating push (i.e., push $e(u)$) on uv makes $e(v)$ positive but $e(u) = 0$. At worst, we need to add $h(v)$ to $\Phi(x, h)$ and subtract $h(u)$ from $\Phi(x, h)$. By the choice of uv, $h(u) = h(v) + 1$, so $\Phi(x, h)$ decreases by at least 1. We have argued that Φ is non-negative throughout the algorithm, so the number of non-saturating pushes is therefore at most $2|N|^2 + 2|N||A|(2|N| - 1) \leq 4|N|^2|A|$. □

1.8.7 Prove: PFP terminates in $2|N|^2 + 2|N|^2|A| + 4|N|^2|A| = 8|N|^2|A| \approx 8|N|^4$ operations.

This follows from the previous proof.

2 Global Minimum Cut

2.1 Global Minimum Cut

2.1.1 Describe the global min cut problem. What's the motivation for an efficient algorithm?

Given $D = (N, A)$ and capacity c, we want to find a global minimum cut, i.e., partition N into two disjoint sets S and S', where $c(\delta(S)) := \{c_{uv} : u \in S, v \in S'\}$ is minimized.

Fix $s \in N$. Observe it must reside on either side of a global min cut.

2.1.2 Define: X, t-cut, s-cut.

Given $X \subseteq N$ and $t \notin X$, an X, t-cut has the form $\delta(S)$ where $X \subseteq S$ and $t \notin S$. In other words, if $\delta(S)$ is an X, t-cut and $uv \in \delta(S)$, then $u \in X \subseteq S$ and $t \in N \setminus S$.

Given $s \in N$, an s-cut has the form $\delta(S)$ where $s \in S$ and $S \neq N$. In other words, if $\delta(S)$ is an s-cut and $uv \in \delta(S)$, then $u \in S$ and $v \in N \setminus S$.

Note that we could reduce the global min cut problem to min s-cut problem. By remark above, if a global min cut contains s, then we are good. Otherwise, reverse all arcs and we will find the global min cut. Thus, we just need to solve min s-cut twice for a fixed s.

2.1.3 Describe the generic algorithm for minimum s-cut.

0. Initialize $X = \{s\}$.
1. While $X \neq N$, pick $t \notin X$, find a min X, t-cut, add t to X.
2. Output the minimum over all cuts found.

2.1.4 Prove: The generic algorithm above solves the minimum s-cut problem.

Let $\delta(S^*)$ be a minimum s-cut. Consider the first time we pick some t not in S^*. Immediately before this step, $X \subseteq S^*$. The algorithm gives us a minimum X, t-cut $\delta(\bar{S})$. But $\delta(S^*)$ is also an X, t-cut (thus $c(\delta(S^*)) \geq c(\delta(\bar{S}))$) and $\delta(\bar{S})$ is also an s-cut (thus $c(\delta(S^*)) \leq c(\delta(\bar{S}))$), so they have the same capacity. □

2.2 Hao-Orlin

2.2.1 Define: X-preflow.

An X-preflow is a flow where every node not in X has a non-negative excess; nodes not in X are allowed to have negative excesses.

2.2.2 What does it mean for height h to be compatible with an X-preflow?

Height h are compatible with an X-preflow if

1. $h(v) = |N|$ for all $v \in X$.
2. $h(t) \leq |X| - 1$.
3. $h(v) \geq h(u) - 1$ for all $uv \in A(D')$.

2.2.3 Define: level, cut level.

A level k, denoted $H(k)$, consists of all nodes with height k.

A cut level is a level K where no arc goes from $H(k)$ to $H(k-1)$ in D'.

2.2.4 Prove: If $\delta(S)$ is an X,t-cut with $\delta_{D'}(S) = \varnothing$ and $e(v) = 0$ for all $v \in N \setminus (S \cup \{t\})$ then $\delta(S)$ is a minimum X,t-cut.

Take any X,t-cut $\delta(S)$. Note the flow out of S is equal to the flow into $N \setminus S$. The net flow out of S is $x(\delta(S)) - x(\delta(\bar{S})) \leq x(\delta(S)) \leq c(\delta(S))$. The net flow into $N \setminus S$ is equal to

$$\sum_{v \in N \setminus S} x(\delta(\bar{v})) - x(\delta(v)) = \sum_{v \in N \setminus S} e(v) \geq e(t).$$

Since $\delta_{D'}(S) = \varnothing$, all out-arcs of S are full and all in-arcs of S are empty, so
$x(\delta(S)) - x(\delta(\bar{S})) = c(\delta(S))$.

Next, $e(v) = 0$ for all $v \in N \setminus (S \cup \{t\})$ tells us

$$\sum_{v \in N \setminus S} e(v) = e(t) + \sum_{v \in N \setminus (S \cup \{t\})} e(v) = e(t) + 0 = e(t).$$

Consider the flow f. Since $e(t) \leq f \leq c(\delta(S))$ for every f and we see that for this particular S we have $e(t) = c(\delta(S))$, S achieves the minimum (lower bound) and hence $\delta(S)$ is a min X,t-cut. $\square$

2.2.5 Prove: If ℓ is a cut level and $e(v) = 0$ for all v with $h(v) < \ell$, except t, then $\{v : h(v) \geq \ell\}$ is a min X,t-cut.

Consider $S := \{v : h(v) \geq \ell\}$, the set of nodes at or above level ℓ. If ℓ is a cut level, there is no arc going from level ℓ to level $\ell - 1$ in D', so $\delta(S)$ is an X,t-cut with $\delta_{D'}(S) = \varnothing$. Combine this with $e(v) = 0$ for all v with $h(v) < \ell$ except t, we see that $\delta(S)$ is indeed a minimum X,t-cut. $\square$

2.2.6 Describe the Hao-Orlin algorithm.

0. Initialization.
   a. Initialize $X = \{s\}$ and pick $t \in N \setminus X$.
   b. Initialize $h(s) = |N|$ and $h(v) = 0$ for all another $v \in N \setminus \{s\}$.
   c. Initialize $\ell = |N| - 1$.
   d. Send as much flow out of s as possible.
1. Loop. While $X \neq N$,
   a. *PFP.* While there exists $u \in N \setminus \{s,t\}$ such that $e_u > 0$ and $h(v) < \ell$:

```
        i. If there exists $uv \in A(D')$ where $h(v) = h(u) - 1$, push $\min\{r_{uv}, e_u\}$ on $uv$.
        ii. Otherwise,
            i. If $v$ is the only node with $h(v)$, do not relabel. Instead, set $\ell = h(v)$.
            ii. If we want to relabel $v$ to $\ell$, reset $\ell = |N| - 1$.
            iii. Otherwise, relabel as before.
    b. *Store.* When no node satisfies $e(v) = 0$ and $h(v) < \ell$, store the cut $\{v : h(v) \geq \ell\}$.
    c. *Reset.*
        i. Add $t$ to $X$, set $h(t) = |N|$.
        ii. Send as much flow out of $t$ as possible.
        iii. Pick a new $t$ with lowest height.
        iv. Reset cut level by setting $\ell = |N| - 1$.
2. Return the the minimum cut amongst all stored cuts.
```

2.3 Correctness of Hao-Orlin

2.3.1 Prove: The non-empty levels less than $|N|$ are consecutive.

Initially, $X = \{s\}$ is at level $|N|$ and everything else is at level 0, so the claim is trivially true at initialization. We do not relabel v when v is the only node of height h; this keeps non-empty levels consecutive. Transitioning to a new iteration, we move t with lowest height to X meaning the non-empty levels remain consecutive. □

2.3.2 Prove: The X-preflow and height h are always compatible.

$h(v) = |N|$ for all $v \in X$ follows from the algorithm. $h(v) \geq h(u) - 1$ for all $uv \in A(D')$ follows from PFP during an iteration. At the end of each iteration, we move t to X and push all flow out of t. Thus, we have $h(v) \geq h(u) - 1$ for all $uv \in A(D')$.

It remains to show that $h(t) \leq |X| - 1$. Initially, $|X| = 1$ and $h(t) = 0 \leq 0 = |X| - 1$. When we move t to X, the next t, call it t', has height $h(t)$ or $h(t) + 1$, since non-empty levels are consecutive. Originally, $h(t) \leq |X| - 1$. We add 1 to $|X|$ after the move and might add 1 to $h(t)$, so this inequality is true after this move.

2.3.3 Prove: $h(v) \leq |N| - 2$ for all $v \notin X$.

Recall $h(t) \leq |X| - 1$. There are $|N| - |X| - 1$ node not in $X \cup \{t\}$. Since the non-empty levels are consecutive, the highest level not in X is at most $(|X| - 1) + (|N| - |X| - 1) = |N| - 2$. □

2.3.4 Prove: ℓ is always a cut level.

When $\ell = |N| - 1$, level ℓ is empty, so it is automatically a cut level. We change ℓ to something else when we want to relabel v, but v is the only node with its height. We want to relabel v because $e(v) > 0$ and no neighbour of v is one level below v in D'. Thus, no arcs goes from $h(v)$ to level $h(v) - 1$. Thus, $h(v)$ is a cut level and we can set $\ell = h(v)$. $\square$

2.3.5 The stored cuts in each iteration are minimum X, t-cuts.

It suffices to show that ℓ is always a cut level, because then by the previous corollary, Hao-Orlin produces a min s-cut. Indeed, by the last proof, ℓ is a cut level. $\square$

2.4 Global Min Cut in Undirected Graphs (Karger's Algorithm)

2.4.1 Describe Karger's algorithm. What's the intuition?

To find the global min cut in undirected graphs, we pick one edge at random, contract it, and keep track of vertices each contracted vertex represents. Do this until two vertices remains and output the cut represented by these two vertices.

To make edges with small capacities more likely to survive, we set probability of an edge gets selected proportional to its capacity.

The algorithm is as follows. While $|V| > 2$, pick uv with probability $\frac{c_{uv}}{\sum_{e \in E} c_e}$ and contract uv. Repeat.

2.4.2 Prove: Let $\delta(S^*)$ be a global min cut. The probability that the algorithm produces $\delta(S^*)$ is $\geq \frac{1}{\binom{|V|}{2}}$.

Consider the probability we pick an edge in $\delta(S^*)$ in the first step. The denominator is $\sum_{e \in E} c_e$ and the numerator is $c(\delta(S^*))$.

Consider the cuts of the form $\delta(\{v\})$ for $v \in V$. Each edge uv appears in two such cuts $\delta(\{u\})$ and $\delta(\{v\})$. Since $\delta(S^*)$ is a global minimum cut,

$$\sum_{e \in E} c_e = \frac{1}{2} \sum_{v \in V} c(\delta(\{v\})) \geq \frac{1}{2} \sum_{v \in V} c(\delta(S^*)) = \frac{1}{2}|V| c(\delta(S^*))$$

So the probability that an edge in $\delta(S^*)$ is selected is

$$\frac{c(\delta(S^*))}{\sum_{e \in E} c_e} \leq \frac{c(\delta(S^*))}{\frac{1}{2}|V| c(\delta(S^*))} = \frac{1}{\frac{1}{2}|V|} = \frac{2}{|V|}.$$

The probability that $\delta(S^*)$ survives the first contraction is thus at least $1 - \frac{2}{|V|}$.

Now suppose we have contracted k edges and $\delta(S^*)$ is still intact. We have $|V| - k$ vertices left and say the graph is $G' = (V', E')$. We want to find the probability of selecting an edge in $\delta(S^*)$.

The numerator is $c(\delta(S^*))$; the denominator is $\sum_{e \in E'} c_e$. Now, each node represents several nodes (because of contractions):

$$\sum_{e \in E'} c_e = \frac{1}{2} \sum_{v \in V'} c(\delta(v)) \geq \frac{1}{2} \sum_{v \in V'} c(\delta(S^*)) = \frac{1}{2}(|V| - k) c(\delta(S^*)).$$

So, the probability is at most

$$\frac{c(\delta(S^*))}{\frac{1}{2}(|V| - k)(c(\delta^*))} = \frac{2}{|V| - k}.$$

The probability that $\delta(S^*)$ survives this contraction is at least

$$1 - \frac{2}{|V| - k}.$$

The largest possible k is $|V| - 3$ (since we finish when we have 2 vertices, so in the last step, we have $3 = |V| - (|V| - 3)$ vertices left).

Overall, the probability that $\delta(S^*)$ survives all contractions is

$$\geq \prod_{k=0}^{|V|-3} \left(1 - \frac{2}{|V| - k}\right) = \prod_{k=0}^{|V|-3} \frac{|V| - k - 2}{|V| - k} = \frac{2}{|V|(|V| - 1)} = \frac{1}{\binom{|V|}{2}}. \quad \square$$

We expect the algorithm to produce a minimum cut if it runs $\binom{|V|}{2} \sim |V|^2$ times.

2.4.3 Prove: The probability that that algorithm produces $\delta(S^*)$ after $k|V|^2$ runs is at least $1 - e^{-2k}$ where $k \geq 1$.

We use $1 - x \leq e^{-x}$. The probability of failure is at most

$$\left(1 - \frac{1}{\binom{|V|}{2}}\right)^{k|V|^2} \leq \left(1 - \frac{1}{|V|^2}\right)^{k|V|^2} \leq \left(e^{\frac{-2}{|V|^2}}\right)^{k|V|^2} = e^{-2k}. \quad \square$$