Swiss Winter School on Theoretical Computer Science 2023

Fast Graph Algorithms Using Optimization and Data Structures

R. Kyng *Problem Set 1, v.0 — Monday, Jan 30th*

Problem: An ℓ_1-Interior Point Method for Maximum Flow

Consider an undirected graph $G = (V, E)$ with capacities $\boldsymbol{c} \in \mathbb{R}_+^E$, as usual with n vertices and m edges. We will assume $m > 10$ and that the entries of $\boldsymbol{c}$ are positive integers and $\|\boldsymbol{c}\|_1 \leq m^{10}$. Let $s, t \in V$ be a source and sink vertex respectively. Assume we are also given the maximum flow value $0 < F \leq m^{10}$ (F is an integer) that can be routed between s and t.

Now we add a new special edge from $\tilde{e}$ between s and t with no capacity limit. We then define the extended edge set $\tilde{E} = E \cup \{\tilde{e}\}$. We pick the orientation of $\tilde{e}$ such that $\mathbf{1}_{\tilde{e}}^\top \boldsymbol{f} = \boldsymbol{f}(\tilde{e})$ measures the amount of flow going from s to t on edge $\tilde{e}$. Let $\boldsymbol{B}$ be the edge-vertex incidence matrix of $\tilde{E}$. Ultimately, we want to find a flow $\boldsymbol{f} \in R^E$ that routes F units of flow from s to t without using the edge $\tilde{e}$. We will do this by setting up a linear program that asks us to find a flow $\boldsymbol{f} \in R^{\tilde{E}}$ that routes F units of flow from s to t without using the edge $\tilde{e}$.

We can now write this as a linear program that asks us to minimize the flow on $\tilde{e}$, subject to the constraints $\boldsymbol{B}\boldsymbol{f} = F(-\mathbf{1}_s + \mathbf{1}_t)$ and for all $e \in E$ that $-\boldsymbol{c}(e) \leq \boldsymbol{f}(e) \leq \boldsymbol{c}(e)$.

$$\begin{aligned} \min_{\boldsymbol{f} \in R^{\tilde{E}}} \quad & \boldsymbol{f}(\tilde{e}) \\ \text{s.t. } \boldsymbol{B}\boldsymbol{f} &= F(-\mathbf{1}_s + \mathbf{1}_t) \\ \text{and for all } e &\in E \\ -\boldsymbol{c}(e) \leq \boldsymbol{f}(e) &\leq \boldsymbol{c}(e) \end{aligned} \tag{1}$$

Now we are going to consider a *barrier program* which is an variant of the above program but using barrier functions that make it suitable for optimizing with an interior point method. We will consider an inequality constraint set that also constrains $\boldsymbol{f}(\tilde{e})$. We denote this capacity constrained flow set by

$$\mathcal{I} = \left\{ \boldsymbol{f} \in \mathbb{R}^{\tilde{E}} : \boldsymbol{f}(\tilde{e}) > 0 \text{ and for all } e \in E. -\boldsymbol{c}(e) < \boldsymbol{f}(e) < \boldsymbol{c}(e) \right\}$$

Then, we define a barrier $B : \mathcal{I} \to \mathbb{R}$ by

$$B(\boldsymbol{f}) = \sum_{e \in E} -\log\left(1 - \frac{\boldsymbol{f}(e)}{\boldsymbol{c}(e)}\right) - \log\left(1 + \frac{\boldsymbol{f}(e)}{\boldsymbol{c}(e)}\right)$$

We further define our overall potential $\Phi(\boldsymbol{f}) = 10m \log(\boldsymbol{f}(\tilde{e})) + B(\boldsymbol{f})$. We can now introduce our barrier program:

$$\begin{aligned} \min_{\boldsymbol{f} \in \mathcal{I}} \quad & \Phi(\boldsymbol{f}) \\ \text{s.t. } \boldsymbol{B}\boldsymbol{f} &= F(-\mathbf{1}_s + \mathbf{1}_t) \end{aligned} \tag{2}$$

Part A: The Potential Function: Initialization and End-Goal. We would now like to begin understanding the Programs (1) and (2) better. First, we will try understand why solutions to Program (1) give us a maximum flow, and second we will investigate how a feasible solution $\boldsymbol{f}$ for Program (2) with a small potential value $\Phi(\boldsymbol{f})$ give us an appproximate maximum flow. Finally, we will see that there is an easy way to get a reasonable feasible starting point $\boldsymbol{f}_0$ for Program (2). Later we will see how to improve this solution to get a maximum flow.

1. Is Program (1) a convex optimization program? Is Program (2) a convex program?
2. Prove that any optimal solution $\boldsymbol{f}^*$ for the Program (1) routes F units of flow from s to t on the edges of E and has $\boldsymbol{f}(\tilde{e}) = 0$.
3. Prove that if $\boldsymbol{B}\boldsymbol{f} = F(-\mathbf{1}_s + \mathbf{1}_t)$, and $\boldsymbol{f} \in \mathcal{I}$, and $\Phi(\boldsymbol{f}) \leq -10m \log m$ then $\boldsymbol{f}(\tilde{e}) \leq 1/m$.
4. Prove that for $\boldsymbol{f}_0 = F\mathbf{1}_{\tilde{e}}$ we have $\boldsymbol{B}\boldsymbol{f}_0 = F(-\mathbf{1}_s + \mathbf{1}_t)$, and $\boldsymbol{f}_0 \in \mathcal{I}$, and $\Phi(\boldsymbol{f}_0) \leq 100m \log m$.

Part B: IPM Progress Using Updates. In Part A of this homework problem, we saw how to find a starting point for minimizing $\Phi(\boldsymbol{f})$ in Program (2). Now, we would like to understand how to improve the solution. To do this, we will think about Taylor series approximations of Φ.

1. Given $\boldsymbol{f} \in \mathcal{I}$, define $\boldsymbol{l} \in \mathbb{R}^{\tilde{E}}$ by $\boldsymbol{l}(\tilde{e}) = 1/\boldsymbol{f}(\tilde{e})$ and

$$\boldsymbol{l}_{\boldsymbol{f}}(e) = \frac{1}{\min(\boldsymbol{c}(e) - \boldsymbol{f}(e), \boldsymbol{c}(e) + \boldsymbol{f}(e))} \tag{3}$$

   for all $e \in E$. We define a corresponding diagonal matrix $\boldsymbol{L}_{\boldsymbol{f}} \in \mathbb{R}^{\tilde{E} \times \tilde{E}}$ by $\boldsymbol{L}_{\boldsymbol{f}}(e, e) = \boldsymbol{l}_{\boldsymbol{f}}(e)$ for all $e \in \tilde{E}$. Prove that if $\|\boldsymbol{L}_{\boldsymbol{f}}\boldsymbol{\delta}\|_\infty \leq 1/2$ then $\boldsymbol{f} + \boldsymbol{\delta} \in \mathcal{I}$ and

$$\Phi(\boldsymbol{f} + \boldsymbol{\delta}) \leq \Phi(\boldsymbol{f}) + \boldsymbol{\nabla}_{\boldsymbol{f}}\Phi(\boldsymbol{f})^\top \boldsymbol{\delta} + 4\|\boldsymbol{L}_{\boldsymbol{f}}\boldsymbol{\delta}\|_1^2.$$

2. Prove that if for some $\kappa > 1$ we have $\|\boldsymbol{L}_{\boldsymbol{f}}\boldsymbol{\delta}\|_1 \leq \kappa$ and $\boldsymbol{\nabla}_{\boldsymbol{f}}\Phi(\boldsymbol{f})^\top \boldsymbol{\delta} = -1$ then

$$\Phi(\boldsymbol{f} + \frac{1}{8\kappa^2}\boldsymbol{\delta}) \leq \Phi(\boldsymbol{f}) - \frac{1}{16\kappa^2}.$$

Part C: The Update. In Part B, we saw a set of sufficient conditions for an update $\boldsymbol{\delta}$ to improve the potential function value. Of course, if we use this to optimization $\Phi(\boldsymbol{f})$ by moving from $\boldsymbol{f}$ to $\boldsymbol{f} + \delta$, then if we require that $\boldsymbol{B}\boldsymbol{f} = F(-\mathbf{1}_s + \mathbf{1}_t)$ and $\boldsymbol{B}(\boldsymbol{f} + \boldsymbol{\delta}) = F(-\mathbf{1}_s + \mathbf{1}_t)$, then we must have $\boldsymbol{B}\boldsymbol{\delta} = \mathbf{0}$, i.e. the update should be a circulation. In this part, we will show that in fact a good circulation $\boldsymbol{\delta}$ exists for lowering the value of Φ, and we can phrase the task of finding $\boldsymbol{\delta}$ as an optimization problem. Finally, we will put everything together and argue that if we can find the update circulations, we can compute a maximum flow in a reasonable amount of time.

1. Consider the following optimization program:

$$\begin{aligned} \min_{\boldsymbol{\delta} \in \mathbb{R}^{\tilde{E}}} \quad & \|\boldsymbol{L}_{\boldsymbol{f}}\boldsymbol{\delta}\|_1 \\ \text{s.t. } & \boldsymbol{B}\boldsymbol{\delta} = \mathbf{0} \\ & \boldsymbol{\nabla}_{\boldsymbol{f}}\Phi(\boldsymbol{f})^\top \boldsymbol{\delta} = -1 \end{aligned} \tag{4}$$

   Is this a convex optimization program?

2. Prove that the value γ^* of Program (4) is bounded by $\gamma^* \leq O(1)$.

   Hint: Consider vectors of the form $\bar{\boldsymbol{\delta}} = \alpha(\boldsymbol{f}^* - \boldsymbol{f})$ for some scalar α, where $\boldsymbol{f}^*$ is an optimal solution to Program (1).

3. Prove that there exists an optimal solution $\boldsymbol{\delta}^*$ for Program (4) which is supported on a single simple cycle.

4. Suppose you have access to the following subroutines:

   - An algorithm that computes a probabilistic low stretch spanning tree of any graph G with m edges in time $\widetilde{O}(m)$. Recall that a probabilistic low stretch tree is a random sub-tree T s.t. for every edge of G, the expected stretch of the edge w.r.t. T is $\widetilde{O}(1)$.
   - An algorithm RoundFlow($\boldsymbol{f}$) that given a flow $\boldsymbol{f} \in \mathcal{I}$ with $\boldsymbol{B}\boldsymbol{f} = F(-\mathbf{1}_s + \mathbf{1}_t)$ returns an *integral* flow[1] $\hat{\boldsymbol{f}}$ with $-\boldsymbol{c}(e) \leq \hat{\boldsymbol{f}}(e) \leq \boldsymbol{c}(e)$, $\hat{\boldsymbol{f}}(\tilde{e}) = 0$ and $\boldsymbol{B}\hat{\boldsymbol{f}} = \hat{F}(-\mathbf{1}_s + \mathbf{1}_t)$ where $\hat{F} \geq F - \boldsymbol{f}(\tilde{e}) - 10$. The routine takes time $T_{\text{RoundFlow}} = \widetilde{O}(m)$.

   Describe a randomized algorithm for solving our undirected maximum flow problem w.h.p. in time $\widetilde{O}(m^2)$ using the IPM, low-stretch spanning trees, the RoundFlow algorithms, and Ford-Fulkerson with basic path augmentation (i.e. no blocking flows). You should not need to use other algorithms such as Dinic's max flow algorithm or data structures like Link-Cut trees.

Part D: Stability. In the previous problem, we started analyzing the update Program (4), and we will start to see why this might be possible, by showing that an update $\boldsymbol{\delta}$ with small norm $\|\boldsymbol{L}_{\boldsymbol{f}}\boldsymbol{\delta}\|_1$ can only cause the lengths $\boldsymbol{l}_{\boldsymbol{f}+\boldsymbol{\delta}}$ to change significantly in very few entries.

1. Consider an update $\boldsymbol{\delta}$ such that $\|\boldsymbol{L}_{\boldsymbol{f}}\boldsymbol{\delta}\|_1 \leq 1/2$, and define $\boldsymbol{s} = \boldsymbol{L}_{\boldsymbol{f}}^{-1}\boldsymbol{l}_{\boldsymbol{f}+\boldsymbol{\delta}}$. Let $\mathcal{U} = \left\{e : |\boldsymbol{s}(e) - 1| > 1/m^{o(1)}\right\}$. Prove that $|\mathcal{U}| \leq m^{o(1)}$.
2. Bonus: Analyze stability of the lengths during a sequence of updates. *Hint: a bucketing scheme may help.*
3. Bonus: Analyze stability of the gradient. What is the right notion of approximation?

Bonus Exercise: Tree data structures

We want to consider a simplified model of our data structures to understand the basics of how we can use a link-cut tree-based data structure to maintain our flow during the IPM iterations.

Assume a min-ratio cycle data structure to support the following operations:

- Initialize($G, \hat{\boldsymbol{g}}^{(0)}, \hat{\boldsymbol{L}}^{(0)}$) : initially, the data structure is given the underlying graph G and the initial approximate gradients and lengths on the edges of G. This also initializes an associated dynamic spanning tree T, whose edges will always be a subset of the edges of G.
- Update($\hat{\boldsymbol{g}}^{(t)}, \hat{\boldsymbol{L}}^{(t)}$) : the t-th update replaces the current gradient and lengths by $\hat{\boldsymbol{g}}^{(t)}$ and $\hat{\boldsymbol{L}}^{(t)}$ (assume a sparse representation can be used, and only specify entries that should be updated). This also returns a list of edges E_{add} that need to be added to T and edges E_{delete} that need to be removed from T. It is guaranteed that T will remain a spanning tree after the update. The amortized number of edges of T that change per update is $m^{o(1)}$.
- Query() : returns an off-tree edge e (w.r.t. the current tree T) and a scaling $\alpha \in R$ such that the cycle given by sending α units of flow forward along e and backwards around its tree path in T (approximately) minimizes (4) with respect to $\hat{\boldsymbol{g}}^{(t)}$ and $\hat{\boldsymbol{L}}^{(t)}$.
- Assume that if $\hat{\boldsymbol{g}}^{(t)} \approx \boldsymbol{g}^{(t)}$ and $\hat{\boldsymbol{L}}^{(t)} \approx \boldsymbol{L}^{(t)}$ (for the lengths, this means coordinate-wise $1 \pm 1/m^{o(1)}$ factor approximation), then the tree cycle for edge e associated with T has value $m^{o(1)}$ in Program (4) and still satisfies the gradient condition (this is slightly unrealistic, but very close the truth).

[1] An integral flow is a flow with integer entries.

For this exercise, you should ignore gradient maintenance: Just assume the gradient vector is maintained correctly and for free by the data structure.

Assume we have another data structure MaintainTree which can support the following operations:

- Initialize(V, T) : Given a spanning tree T with edges F on vertex set V, initialize the data structure. The edges should have an orientation (but this is not part of the relevant to the property of being a spanning tree). Maintain a vector $\boldsymbol{t} \in \mathbb{R}^F$ of real numbers, one for each edge in T. Initially all entries of $\boldsymbol{t}$ are set to zero. Time is $\widetilde{O}(m)$.
- Update$(E_{\text{add}}, E_{\text{delete}})$: Remove edges E_{delete} from T and add edges E_{add} to T. The edges must maintain that T is a spanning tree of V. This changes vector $\boldsymbol{t}$: new entries are set to zero. Time per edge is $\widetilde{O}(1)$.
- QueryMax(u, v) : Given vertices $u, v \in V$, returns the maximum entry in $\boldsymbol{t}$ (index and value) on the path between u and v. Time per query is $\widetilde{O}(1)$.
- AddUnsigned(u, v, δ) : Given vertices $u, v \in V$, add $\delta \in \mathbb{R}$ to every entry of $\boldsymbol{t}$ corresponding to edges on the u-v path in T. Time for the operation is $\widetilde{O}(1)$.
- AddSigned(u, v, δ) : Given vertices $u, v \in V$, add $\pm\delta$ to every entry of $\boldsymbol{t}$ corresponding to edges on the u-v path in T: Add $+\delta$ to the edge if the tree path traverses the edge in the direction of its orientation. Add $-\delta$ to the edge if the tree path traverses the edge in the opposite direction of its orientation. Time for the operation is $\widetilde{O}(1)$.
- You may also assume operations for QuerySumSigned(u, v) and QuerySumUnsigned(u, v) : (self-explanatory), but I don't think you should need them. Time for the operation is $\widetilde{O}(1)$.

The goals of the exercise:

- Describe how to maintain the solution flow vector $\boldsymbol{f}^{(t)}$ during the IPM under updates given by tree cycle flows. Overall time should be $m^{1+o(1)}$.
- Describe how to maintain the lengths $\hat{\boldsymbol{L}}^{(t)} \approx \boldsymbol{L}^{(t)}$ with a coordinate-wise $1 \pm 1/m^{o(1)}$ factor approximation. Overall time should be $m^{1+o(1)}$.