University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Schneider

Advanced Algorithms
Sample Solution Problem Set 10

Issued: Monday, July 15, 2019

Exercise 1: Evaluating Congestion Approximators

As we have seen in the lecture, we can construct a m-congestion approximator based on a maximum spanning tree T as follows ($m := |E|$). For each edge $e \in T$ let S_e be the cut induced by e in the graph. Then we set $R_{e,v} = 1/c_{S_e}$ for all $v \in S_e$ and $R_{e,v} = 0$ for all $v \notin S_e$, where c_{S_e} is the sum of capacities of edges going over the cut S_e. The entries $R_{e,v}$ form a $(n-1) \times n$-matrix R. Show that for $x \in \mathbb{R}^n, y \in \mathbb{R}^{n-1}$ we can compute Rx and $R^\top y$ in $O(n)$. Assume the capacities of the cuts $c_{S_e}, e \in T$ are known.

Sample Solution

Assume nodes are numbered from 1 to n and edges from 1 to m. The e-th entry of the vector Rx can be computed as

$$(Rx)_e = \sum_{v \in S_e} \frac{x_v}{c_{S_e}}.$$

Let $r \in V$ be an arbitrary node that we designate as root of T. For any edge $e = \{u, w\} \in T$, where u is closer to the root r, let the cut S_e be the nodes in the subtree rooted at w (instead of the other way around).
As the c_{S_e} are already known, this means that all we have to do to compute $(Rx)_e$ is to sum up x_v for all nodes of the subtree of T rooted at the "lower" endpoint w of $e = \{u, w\}$. We can compute any of these sums with a simple recursive approach.

Algorithm 1 $\textsc{SumSubtree}(e = \{u, w\})$ ▷ *global dictionary* `memo`

```
if memo(e) ≠ ⊥ then                                      ▷ partial result not yet computed
    memo(e) ← x_v + Σ_{child v of w} SumSubtree({w, v})   ▷ compute partial result
return memo(e)
```

Since we recurse at most once for each edge e before the partial result is globally available, the runtime is $O(n)$. Furthermore, we can compute the v-th entry of $R^\top y$ as follows

$$(R^\top y)_v = \sum_{e \in T:\, v \in S_e} \frac{y_e}{c_{S_e}}.$$

The node v is exactly in all subtrees S_e for $e = \{u, w\}$ if u is an ancestor of v, or $u = v$. Therefore, the above formula sums all y_e/c_{S_e} for all $e \in T$, where e is on the path from v to the root r. Again we can compute these cheaply with a recursive approach:

Algorithm 2 SUMPATH(T, v) ▷ *global dictionary* `memo`

```
if v = r then return
u ← ancestor of v in T
if memo(u) ≠ ⊥ then                                      ▷ partial result not yet computed
    memo(u) ← SumPath(T, u) + y_{v,u}/c_{S_{v,u}}       ▷ compute partial result
return memo(v)
```

Again we recurse at most once for each node before partial results are globally available. Thus the total running time is $O(n)$.

Exercise 2: Analysis of the Gradient Descent Procedure

In the lecture we saw that we can reduce the max flow problem to a continuous, unrestricted optimization problem that we solved with the gradient descent method.
Show that one step of gradient descent requires $\tilde{O}(m)$ time and one multiplication with R and another one with $R^\top$ (where R is the congestion approximator used in the procedure).

Hint: Use the following chain rule for gradients: for $h(x) := g(Ax)$, we have $\nabla h(x) = A^\top \cdot \nabla g(Ax)$.

Sample Solution

The optimzation problem we obtained in the lecture is as follows

$$\min_{\text{flow} f} \gamma(f) := \|C^{-1}f\|_\infty + 2\alpha \|R(b - Bf)\|_\infty$$

We approximate the supremum norm with the lmax function which is (for some $x \in \mathbb{R}^d$) defined as

$$\text{lmax}(x) := \ln\Big(\sum_{i=1}^{d} e^{x_i} + e^{-x_i}\Big).$$

The gradient $\nabla\text{lmax}(x)$ is given (coordinate-wise) by

$$\big(\nabla\text{lmax}(x)\big)_i = \frac{e^{x_i} - e^{-x_i}}{\sum_{j=1}^{d} e^{x_j} + e^{-x_j}}.$$

We obtain the following approximate optimization problem

$$\min_{\text{flow} f} \phi(f) := \text{lmax}\big(C^{-1}f\big) + 2\alpha \cdot \text{lmax}\big(R(b - Bf)\big).$$

In order to optimize ϕ with gradient descent we have to compute the gradient $\nabla\phi$. Let $x := C^{-1}f$ and let $y := R(b - Bf)$. Then according to the chain rule from the hint we have

$$\nabla\phi(f) = \nabla\text{lmax}(x) + 2\alpha B^\top R^\top \nabla\text{lmax}(y).$$

In summary we have to compute x and y where the first takes $O(m)$ and the latter takes $O(m)$ and a multiplication with R (since the vector f is of dimension m and C, B have $O(m)$ entries). Additionally we have to compute the sums in the denominators of $\big(\nabla\text{lmax}(x)\big)_i$ and $\big(\nabla\text{lmax}(y)\big)_i$ just once, which takes $O(m)$, hence the computation of $\nabla\text{lmax}(x)$ and $\nabla\text{lmax}(y)$ takes $O(m)$. Finally we require a multiplication $R^\top\nabla\text{lmax}(y)$.