Assignment: Assignment 2

Pass the exercise

You received 0 reviews and 0 likes
(/learn/approximation-algorithms-part-2/peer/tyX7r/assignment-2/submit)

Review 3 classmates

0/3 reviews completed
(/learn/approximation-algorithms-part-2/peer/tyX7r/assignment-2/give-feedback)

Instructions (/learn/approximation-algorithms-part-2/peer/tyX7r/assignment-2)

My submission (/learn/approximation-algorithms-part-2/peer/tyX7r/assignment-2/submit)

Review classmates (/learn/approximation-algorithms-part-2/peer/tyX7r/assignment-2/give-feedback)

Discussions (/learn/approximation-algorithms-part-2/peer/tyX7r/assignment-2/discussions)

Your work is submitted.
It will now be reviewed by your classmates. When your feedback is ready, we'll email you. In the meantime, you should review classmates' submissions.

Start Reviewing (/learn/approximation-algorithms-part-2/peer/tyX7r/assignment-2/give-feedback)

A Primal-Dual Algorithm for the Shortest Path Problem.

February 13, 2016

Shareable Link (https://www.coursera.org/learn/approximation-algorithms-part-2/peer/tyX7r/assignment-2/review/DG6LltJhEeWx_BI9PC2FcQ)

A Primal-Dual Algorithm for the Shortest Path Problem.

In this exercise, we propose to design a primal-dual algorithm for the shortest path problem.

The shortest path problem is as follows: given a connected graph $G = (V, E)$, a cost function on the edges $w : E \to R_+$, and two vertices s and t, find the minimum-cost path that connect s to t in G. This problem can be efficiently by Dijkstra's algorithm, however, we can derive a Primal-Dual algorithm that efficiently computes the shortest path.

Throughout the exercise we define $\mathcal{S} = \{S \mid S \subset V \text{ and } s \in S \text{ and } t \notin S\}$ and, for each set $S \subset V$, by $\delta(S)$ the set of edges that have an endpoint in S and an endpoint in $V \setminus S$.

We will consider the following linear program LP for the problem.

$\min \sum_{e \in E} x_e \cdot w(e)$

subject to,

$\forall S \in \mathcal{S}, \quad \sum_{e \in \delta(S)} x_e \geq 1$

$\forall e \in E, \quad x_e \geq 0$

Question 1: What is the dual of this linear program?

We now consider the following primal-dual algorithm.

1. $F \leftarrow \emptyset$
2. $y \leftarrow 0$
3. While there is no path connecting s to t in F:

- Let C be the connected component of the graph $G' = (V, F)$ containing s.
- Increase the dual variable y_C until there exists an edge e' such that $\sum_{S \in \mathcal{S} \,:\, e \in \delta(S)} y_S = w(e')$
- Add the edge e' to F

1. Return a path P in F that connects s to p.

Correctness.

Question 2: In how many iterations of the while loop can a given dual variable be increased?

Question 3: Using Question 2, argue that the algorithm terminates and so, that the output P is a solution to the problem.

Approximation Ratio.

We first want to prove the following Lemma that will be of great help in the proof of the approximation ratio:

Lemma 1. At any step of the algorithm, the set F is a tree that contains s.

We will proceed by induction on the number of edges i added to C.

Question : Prove the case $i = 1$.

Question : Assume that the Lemma holds up to $i - 1$ and prove that it is true for i.

Question 4: Recall a tight lower bound between the value of the optimal fractional solution for the dual val($y*$) and the value of the shortest path between s and t, $P*$.

Question 5: Argue that the solution y is feasible for the dual.

Question 6: Combining Questions 4 and 5, recall a tight lower bound between the value of the solution y and the value of the shortest path between s and t, $P*$.

As usual for primal-dual algorithm, we want to show

$\sum_{e \in P} w(e) \leq \text{val}(y)$.

Question 7: Consider an edge $e \in P$. What is the relationship between $w(e)$ and $\sum_{S \in \mathcal{S}\,:\, e \in \delta(S)} y_S$?

Question 8: Using Question 7, give the relationship between $\sum_{e \in P} w(e)$ and the variables y_S.

Question 9: Using Question 8, give the relationship between $\sum_{e \in P} \sum_{S: e \in \delta(S)} y_S$ and the y_S and $|P \cap \delta(S)|$.

We now want to prove the following. For all S, if $y_S > 0$ then $|P \cap \delta(S)| = 1$.

Assume toward contradiction that this is not the case. Then there exists a set S such that $y_S > 0$ and $|P \cap \delta(S)| \geq 2$. It follows that P crosses $\delta(S)$ multiples times.

Question 10: Using Lemma 1 and the fact that $y_S > 0$, explain the contradiction.

Question 11: Conclude using Questions 6, 9, and 10.

Question 12: Explain why the pruning part (the part where we remove the edges that are not in P) is important for the proof. In which question do we use this?

Let's compare with Dijkstra's algorithm. Recall that Dijkstra algorithm works as follows.

The algorithm starts with $d(i) = w((s, i))$ for each edge (s, i) and $d(i) = \infty$ for each vertex i that is not a neighbor of s.

Moreover, the algorithm starts with a subset $D = \{s\}$ of vertices.

At each step it adds to D the vertex $i \notin D$ that minimizes $d(i)$ and updates the $d(i)$ as follows: for each neighbor $j \notin D$ of i,

$d(j) = \min(d(i) + w((i, j)), d(j))$.

Now, consider the primal-dual algorithm. It starts with a set $F = \emptyset$ and at each step,

increases the dual variable corrsponding to the set of vertices induced by $F \cup \{s\}$ until a constraint becomes tight and adds

the corresponding edge to F.

Question 13: Prove that the edge (s, i) is the first edge added to F if and only if i is the second vertex added to D (the first being s).

We define a notion of time, initially the time is 0 and after we increased a dual variable by ϵ, the time is ϵ.

We now fix a set C_0 that is a connected component of F containing s at some time in the execution of the primal-dual algorithm.

Denote by $a(e)$ the time at which constraint $\sum\limits_{S \in \mathcal{S}\,:\, e \in \delta(S)} y_S = w(e)$ would become tight if we never stop to increase variable y_S (even if some other gets violated).

Moreover, for all $j \notin C_0$, let $l(j) = \min_{(j,i)\in E} a(e)$.

Question 14: Using the $l(j)$, which vertex of $V \setminus C_0$ will be added to C_0 in the next iteration?

Question 15: In the algorithm, we stop increasing variable y_S after we added the new vertex j and in the next iteration we will increase a variable y'_S, where $S' = S \cup \{j\}$. Which edges appear in $\delta(S')$ and not in $\delta(S)$?

Question 16: Explain how to modify the variables $l(k)$ for each vertex $k \notin S'$ after the algorithm added the vertex j to S.

Question 17: Using Question 16 and the definition of $d(j)$, conclude about the order in which the vertices are added to the graph $G' = (V, F)$ by the primal-dual algorithm and to the set D by Dijkstra algorithm.

Complexity:

Question 18: Based on question 13, what is the best known worst-case complexity for the primal-dual algorithm?

Answers

1. **The Dual:**

   $max \sum_{S\in S}\ y_S$
   subject to,

   $\forall e \in E,\quad \sum_{S:e\in\delta(S)} y_S \leq w(e)$

   $\forall S \in S,\quad y_S \geq 0$

2. y_C can by increased only once in the loop.Since for some set S with $C \subset S \subset V$, y_S already becomes tight in that iteration itself, y_C can't be increased further (in any further iterations), otherwise that constraint will get violated and the dual will become in-feasible.

3. At each iteration the algorithm covers the minimum $s - t$ cut $(S, V - S)$ from the set of all such cuts $\delta(S)$ still uncovered and adds the corresponding edge crossing the cut S to the solution set F and the corresponding new vertex (the other end of $e\prime$ now becomes a part of the component containing s). Hence, in each iteration, the component containing the source s expands by one more (new) edge. But there are finite number of (at most $O(|V|^2)$) edges in the graph, hence the while loop must terminate.

   Correctness:

   Since the algorithm has already covered all the minimum $s - t$ cuts before termination, it will end with a shortest path in between s and t.

   Lemma 1. At any step of the algorithm, the set F is a tree that contains s.

   (Proof by induction on the number of edges i added to C).

   Base case: For $i = 1$, L only contains one edge $e\prime$ and hence is trivially a tree (a connected graph with 2 vertices and 1 edge).

   Induction Hypothesis: Let's assume that the Lemma holds up to $i - 1$.

   Induction Step: Now let's prove that it is true for i. When the i^{th} edge $e\prime$ added to the set F, by algorithm steps, $e\prime \in \delta(S)$, which implies that it must have crossed an $s - t$ cut yet to be covered.Now, by induction hypothesis, we had an existing tree in F with $i - 1$ edges (which also means it has \$i\$ vertices), which represent the component containing the source s and the edge $e\prime$ will be chosen such that it has one end in the component containing s and the other end containing the target t. Hence, F remains connected and it now has one extra edge and one additional vertex (the other end was already in F), with total number of vertices after adding the i^{th} edge becomes $i + 1$. Hence, F still remains a tree.

4. By **Weak Duality**, we have $val(y*) \leq val(P*)$.

5. Also, since all the constraints remain satisfied during the iterations ($\forall S,\ y_S \geq 0$ and $\forall e \in E$, $\sum_{S:e\in\delta(S)} y_S \leq w(e)$, only for a few edges $e\prime$ crossing the $s - t$ cuts, these constraints at most become tight), $val(y)$ is feasible for the dual.

6. Combining 4 and 5, we have $val(y) \leq val(y*) \leq val(P*)$, since the optimal dual solution is the maximum achievable value by the dual objective.

7. $e \in P \Leftrightarrow e \in F\ \Leftrightarrow\ \sum_{S:e\in\delta(S)} y_S = w(e)$ (the constraint must be tight for some cut $(S, V - S)$).

8. Hence, $\sum_{e\in P} w(e) = \sum_{e\in P}\ \sum_{S:e\in\delta(S)} y_S$.

9. By exchanging the summations, we have, $\sum_{e\in P} w(e) = \sum_{S:e\in\delta(S)}\ \sum_{e\in P}\ y_S = \sum_S y_S |P \cap \delta(S)|$.

10. **Proof by contradiction**: by our assumption, $\exists$ a set S s.t., $y_S > 0$ and $|P \cap \delta(S)| \geq 2$. It means P contains 2 edges that crossed the same $s - t$ cut $(S, V - S)$ for some S. Now,by Lemma 1 and the algorithm steps, when an edge $e\prime$ crossing the cut $(S, V - S)$ was added to P for the first time when the cut $(S, V - S)$ was considered ($y_S >\ 0$), P still remained a tree. But for the 2nd time, when the same cut $(S, V - S)$ was considered, all its vertices were in the tree (by Lemma 1) that contained the source s, which implies that it would add an edge between two vertices of a tree, by creating a cycle, a contradiction to Lemma 1. Hence, if we have $y_S > 0$ (the corresponding cut was chosen once), we must have $|P \cap \delta(S)| = 1$ (since it must be greater than equal to one, since one edge crossing the cut must be chosen by the algorithm steps).

11. Hence, by 6, 9, 10,
    $val(P) = \sum_{e\in P} w(e) = \sum_S y_S |P \cap \delta(S)| = \sum_{S:y_S>0} y_S |P \cap \delta(S)| +\ \sum_{S:y_S=0} y_S |P \cap \delta(S)| = \sum_{S:y_S>0} y_S.1 +\ \sum_{S:y_S=0} 0.|P \cap \delta(S)|$
    $=\ \sum_{S:y_S>0} y_S + 0 =\ \sum_{S:y_S} y_S = val(y) \leq val(y*) \leq val(P*)$. By optimality of $P*$ it follows that $P = P*$.

12. Since the algorithm may add redundant edges that are not on $s - t$ path, pruning them is important. Otherwise it will in the worst case increase the time complexity of the algorithm to $O(|V|^2)$. It was necessary to show in question 10 that if $y_S > 0$, we must have $|P \cap \delta(S)| = 1$ .If we replace P by F, it's not necessary in this case.

13. i is the second vertex added to D by Dijkstra iff $d(i)$ is the minimum distance vertex from the source $\Leftrightarrow$ the edge (s,i) is the minimum-weight edge crossing the $s-t$ cut $(s, V-s)$ $\Leftrightarrow$ The edge for which the constraint $\sum_{S:e\in\delta(S)} y_S \leq w(e)$ will become tight first will be this min-weight edge (s,i), hence it will be added to the set F first.
14. The vertex next to be added to C_0 is the vertex j that minimizes $l(j)$.
15. The vertex j was not in S, but is in $S\prime$. Now the edges that have one end vertex j can be partitioned into 2 sets: $E_1 = \{(j,k) | k \in S\}$ and $E_2 = \{(j,k) | k \notin S\}$. The cut $\delta(S\prime) = (S\prime, V - S\prime)$ contains all the edges E_2 but the cut $\delta(S) = (S, V-S)$ does not contain these edges, $\delta(S)$ contains the edges E_1.
16. We need to relax $l(k)$ in the following way: $l(k) = min(l(k), l(j) + w((j,k))$.
17. Since Dijkstra also adds the vertex to D in the similar way, the order will be same.
18. If we use min heap (Fibonacci heap) the worst case complexity of the algorithm will be same as Dijkstra $O(E + VlgV)$.

Edit submission

Comments
Visible to classmates

share your thoughts...