TD 10 & 11 - Rounding and Primal Dual algorithms

Indication of hardness: from (*) to (****).

1 Rounding

Exercise 1 - 3/4-approximation of SAT (*)
In class, we gave a $\frac{3}{4}$-approximation algorithm for the maximum satisfiability problem. Give a tight example for this algorithm. In other words, give an instance for which the expected value of the solution returned by the algorithm is $\frac{3}{4}OPT$.

Exercise 2 - Integer Multicommodity Flow (**)
Given a graph $G = (V, E)$ and k pairs (s_i, t_i) (where $s_i, t_i \in V$ for all $k = 1, \dots, k$), our goal is to find a path from s_i to t_i for $i = 1, \dots, k$ so that the maximum *edge congestion* is minimized. Let P_i denote the set of all paths from s_i to t_i. We have seen during the lecture the following linear programming relaxation.

$$\begin{aligned} &\min C \\ &\sum_{p \in P_i} x_p = 1, \quad \text{for all } i = 1, \dots, k, \\ &\sum_{p : e \in p} x_p \leq C, \quad \text{for all } e \in E, \\ &\quad x_p \geq 0. \end{aligned} \tag{P^1_{flow}}$$

However it might have an exponential number of variables. Consider another linear programming relaxation with a polynomial number of variables. In this relaxation, x_{ie} represents the number of paths using edge e.

$$\begin{aligned} &\min C \\ &\sum_{e \in \delta^+(v)} x_{ie} = \sum_{e \in \delta^-(v)} x_{ie}, \quad \text{for all } i = 1, \dots, k \text{ and } v \neq s_i, t_i, \\ &\sum_{e \in \delta^-(s_i)} x_{ie} = \sum_{e \in \delta^+(t_i)} x_{ie} = 1, \quad \text{for all } i = 1, \dots, k, \\ &\sum_{i=1}^{k} x_{ie} \leq C, \quad \text{for all } e \in E \\ &\quad x_{ie} \geq 0. \end{aligned} \tag{P^2_{flow}}$$

Show that relaxations (P^1_{flow}) and (P^2_{flow}) are equivalent in the sense that an optimal solution for (P^1_{flow}) can be converted to an optimal solution for (P^2_{flow}) and vice versa.

Exercise 3 - Minimum Covering Radius (***)
We are considering the following problem. We are given k words on alphabet $0, 1$ and the goal is the determine the word that is the closest from all these words in Hamming distance (for instance these k words might be the same word passing through a channel with loss and the goal is to determine the original word. The Hamming distance $d(u, v)$ the number of bits different between u and v. Formally we have:
Input: k words $S_1, \ldots, S_k$ of length n.
Input: The minimum C such that there exists w such that $d(S_i, w) \leq C$ for every i.

1. Formulate the problem as an ILP.
2. Let x^* be an optimal solution of the fractional relaxation. Let us denote by x_i the variable corresponding to the i-th letter. Now let us set $x_i = 1$ with probability x_i^* and 0 otherwise. Prove that the expected distance from a randomized rounding to the word S_i is at most C for every i.
3. Deduce a $\log(k)$ approximation algorithm.

2 Primal-Dual algorithms

Exercise 4 - The Hungarian Method for the Assignment Problem (***)
Let $G = (V, E)$ be a bipartite graph whose edge costs are nonnegative integers. There is a bipartition $V = (A, B)$ where $|A| = |B| = n$ and the goal is to assign each element in A (e.g. people) to a unique element in B (e.g. tasks) so as to minimize the total cost of the assignment. In other words, we want to find a minimum cost perfect matching between A and B. The goal of this exercise is to study the following primal-dual algorithm for this problem. Let C denote the $n \times n$ cost matrix, where rows are indexed by vertices in A and columns are indexed by vertices in B.

1. For each row in C, decrease each value by the cost of the minimum entry in the row. (Then do the same for each column.) Call the resulting cost matrix $\bar{C}$. Let G_0 denote the subgraph of G that consists of edges in G whose cost in $\bar{C}$ is zero, i.e. $\bar{c}_{ij} = 0$.
2. Find a maximum cardinality matching in G_0. If this matching has size n, terminate the algorithm.
3. Otherwise, find a minimum vertex cover in G_0. Let $A' \subset A, B' \subset B$ denote the vertices in the vertex cover. Note that $|A'| + |B'| < n$.
4. Let $\alpha = \min_{(i,j): i \notin A', j \notin B'} \bar{c}_{ij}$. Subtract α from every row in $\bar{C}$ that is not in A' and add α to each column in B'. Set $C := \bar{C}$ and goto Step 1.

We will now analyze this algorithm.

(a) Apply this algorithm to the following 5 by 5 matrix.

$$\begin{pmatrix} 2 & 3 & 4 & 6 & 8 \\ 5 & 5 & 7 & 2 & 3 \\ 6 & 3 & 1 & 2 & 2 \\ 7 & 5 & 4 & 3 & 6 \\ 8 & 7 & 5 & 3 & 2 \end{pmatrix}$$

(b) Prove that Step 1 of the algorithm does not affect (i.e. change) the optimal assignment.

(c) Show that the maximum matching in Step 2 can be found efficiently.

(d) Show that the minimum vertex cover in Step 3 can be found efficiently.

(e) Prove that the algorithm terminates.

(f) Write the primal and dual linear programs for the assignment problem.

(g) Interpret the above algorithm as a primal-dual algorithm.

(h) Prove that the final solution is a minimum cost perfect matching by providing a dual certificate.

Exercise 5 - Primal-Dual and Dijkstra's Algorithm (*)
Prove that the primal-dual algorithm for shortest s-t-path is equivalent to Dijkstra's algorithm. That is, in each step, it adds the same edge Dijkstra's algorithm would add.

Exercise 6 - Shortest s-t-path Tree (*)
Show that the primal-dual algorithm for shortest s-t-path returns a (possible partial) shortest path tree rooted at s before pruning.

Exercise 7 - Minimum Cost Arborescence (**)
Given a (strongly connected) directed graph $G = (V, A)$ and a root vertex $r \in V$, an *arborescence* is a subset of edges $S \subseteq A$ such for each vertex $v \in V$, S contains a directed path from r to v. Suppose that each edge $ij \in A$ has a cost $c_{ij} \geq 0$. The *minimum cost arborescence problem* is to find an arborescence in G of minimum cost.

(a) Write down the integer program for the minimum cost arborescence problem.

(b) Relax the integrality constraint in the integer program to obtain a linear programming relaxation. Write the dual for this linear program.

(c) Give a primal-dual algorithm for the minimum cost arborescence problem. (Use the same framework as for the s-t-shortest path. For the pruning stage, delete edges in the reverse order they were added.)

(d) Prove that this algorithm is optimal.