Approximation Algorithms	**Dept. of CS, Aarhus University**
Homework 4	
Lecturer: Nguyen Kim Thang	*Student: ??*

1 Primal-Dual Formulation

Formulate and write the relaxed linear programs together with their dual for the following problems. For the last problem (Shortest Super String), no dual is required. Remember to define the variables in the primal. For example, in Vertex Cover, we defined for $u \in V$:

$$x_u = \begin{cases} 1 & \text{if } u \text{ is in the vertex cover,} \\ 0 & \text{otherwise.} \end{cases}$$

Set Cover Given an universe $U = u_1, \dots, u_n$, a collection of subset $\mathcal{S} = \{S_1, \dots, S_k\}$, and a cost function $c : \mathcal{S} \to \mathbb{Q}^+$. A *set cover* is a sub-collection $\mathcal{C}$ of $\mathcal{S}$ that covers all element in U, i.e., for all $u \in U$, $u \in \bigcup_{S \in \mathcal{C}} S$. Find a minimum cost set cover.

Minimum Spanning Tree Given an undirected graph $G(V, E)$ and a weight function $w : E \to \mathbb{Q}^+$. Find a minimum weight spanning tree, i.e., a tree spans all vertices V.

Steiner Tree Given an undirected graph $G(V, E)$, a weight function $w : E \to \mathbb{Q}^+$ and a set of terminal $S \subset V$. Find a minimum weight tree that spans all node in S.

Traveling Salesman tour Given a complete undirected graph $G(V, E)$, a weight function $w : E \to \mathbb{Q}^+$. Find a minimum weight simple cycle that visits all nodes in V.

Shortest Super String Given a set of n strings $\{s_1, \dots, s_n\}$ over a finite alphabet $\sum$. Find a minimum length string t that contains each s_i as a substring.

2 Duality

Solve exercise 12.8 in the *Approximation Algorithms*.

3 Proof

3.1 Set Cover

Define for each set $S \in \mathcal{S}$

$$x_S = \begin{cases} 1 & \text{if } S \text{ is chosen,} \\ 0 & \text{o.w.} \end{cases}$$

We have the following LP primal-dual:

$$\begin{array}{llcll} \min & \sum_{S \in \mathcal{S}} c_S x_S & & & \\ \text{s.t} & \sum_{S: u \in S} x_S & \geq & 1 & \forall u \in \mathcal{U}, \quad (1) \\ & x_S & \geq & 0 & \forall S \in \mathcal{S}. \end{array} \qquad \begin{array}{llcll} \max & \sum_{y_u} y_u & & & \\ \text{s.t} & \sum_{u: u \in S} y_u & \leq & c_S & \forall S \in \mathcal{S} \\ & y_u & \geq & 0 & \forall u \in \mathcal{U}. \end{array}$$

where the constraint (1) in the primal says that each element is covered by at least one set.

3.2 Minimum Spanning Tree

Define for each edge $e \in E$

$$x_e = \begin{cases} 1 & \text{if } e \text{ is chosen,} \\ 0 & \text{o.w.} \end{cases}$$

We have the following LP primal-dual:

$$\begin{array}{rcll} \min \sum_{e \in E} c_e x_e & & & \\ \text{s.t} \sum_{e \in E} x_e & \geq & |V| - 1, & \quad (2) \\ \sum_{e: e=(u,v), u \in S, v \in S} x_e & \leq & |S| - 1 \quad \forall S \subseteq V, & \quad (3) \\ x_e & \geq & 0 \quad \forall e \in E. & \end{array}$$

In the LP, the constraint (2) in the primal ensures that we have at least $|V| - 1$ edges in the solution (any MST has exactly $|V| - 1$ edges). The constraint (3) makes sure there is no cycle by requiring that for any subset S of vertices, there are at most $|S| - 1$ edges between vertices of S in the solution, so the vertices of S are not on a cycle in the solution. Since that is applied for any subsets, the solution contains no cycle and hence it is a forest. As it contains at least $|V| - 1$ edges, it only contains a single connected component, therefore it is a tree.

Rewrite the primal, we get the following LP primal-dual.

$$\begin{array}{rcll} \min \sum_{e \in E} c_e x_e & & & \\ \text{s.t} \sum_{e \in E} x_e & \geq & |V| - 1, & \\ \sum_{e: e=(u,v), u \in S, v \in S} -x_e & \geq & 1 - |S| & \forall S \subseteq V, \\ x_e & \geq & 0 & \forall e \in E. \end{array} \qquad \begin{array}{rcll} \max \quad (|V| - 1)\alpha + \sum_{S \subseteq V} (1 - |S|)\beta_S & & & \\ \text{s.t} \quad \alpha - \sum_{S: u,v \in S} \beta_S & \leq & c_e & \forall e = (u,v) \in S \\ \beta_S & \geq & 0 & \forall S \subseteq V. \\ \alpha & \geq & 0 & \end{array}$$

3.3 Steiner Tree

A LP-formulation can be done similarly as STEINER FOREST in class. Here we present another LP. For any subset $T \subset V$, denote $\delta(T)$ the cut of T. Define for each edge $e \in E$

$$x_e = \begin{cases} 1 & \text{if } e \text{ is chosen,} \\ 0 & \text{o.w.} \end{cases}$$

We have the following LP primal-dual:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t} \quad & \sum_{e: e \in \delta(T)} x_e \geq 1, \quad \forall T \subset V, S \not\subset T \quad (4) \\ & x_e \geq 0 \quad \forall e \in E. \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{T: T \subset V, S \not\subset T} y_T \\ \text{s.t} \quad & \sum_{T: e \in \delta(T), S \not\subset T} y_T \leq c_e \quad \forall e \in E, \\ & y_T \geq 0 \quad \forall T \subset V, S \not\subset T. \end{aligned}$$

where the constraint (4) ensures that all nodes of S are connected. We do not care about cycles since in any minimal solution does not contain cycle.

3.4 Traveling Salesman tour

Define for each edge $e \in E$

$$x_e = \begin{cases} 1 & \text{if } e \text{ is chosen,} \\ 0 & \text{o.w.} \end{cases}$$

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t} \quad & \sum_{e \in E} x_e = |V|, & (5) \\ & \sum_{e: e=(u,v), u,v \in S} x_e \leq |S| - 1 \quad \forall S \subset V, & (6) \\ & x_e \geq 0 \quad \forall e \in E. \end{aligned}$$

where the constraint (5) guarantees that the output has $|V|$ edges, a necessary condition for a tour. The constraint (6) (on proper subset of V) guarantees no cycle in the output graph except the tour itself.

Rewrite the primal, we have the following LP primal-dual:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
\text{s.t} \quad \sum_{e: e \in E} x_e & \geq |V|, \\
\sum_{e: e \in E} -x_e & \geq -|V|, \\
\sum_{e: e=(u,v), u,v \in S} -x_e & \geq 1 - |S| \quad \forall S \subset V, \\
x_e & \geq 0 \quad \forall e \in E.
\end{aligned}
$$

$$
\begin{aligned}
\max \quad |V|(\alpha - \beta) \quad + & \sum_{S \subset V} (1 - |S|)\gamma_S \\
\text{s.t} \quad \alpha - \beta - \sum_{S: S \subset V, u,v \in S} \gamma_S & \leq c_e \quad \forall e = (u,v) \in E, \\
\alpha, \beta & \geq 0 \\
\gamma_S & \geq 0 \quad \forall S \subset V.
\end{aligned}
$$

3.5 Shortest Super String

Assume that the first character index in a string is 1 (instead of 0 as usual in programming). Let N be the sum of all strings' length, i.e., $N = \sum_{i=1}^{n} |s_i|$. Clearly $|t| \leq N$. Let s_{kj} denote the j^{th} character of s_k and t_i is the character at position i in the output string t. Define the following variables. Variable x_{ijk} indicates whether t_i covers the occurrence of s_{kj}. Variable y_i indicates whether a specific location i in the output string is used to cover some character. The objective is to minimize the number of characters used in the cover.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{N} y_i \\
\text{s.t} \quad \sum_{i=1}^{N} x_{ijk} & \geq 1, \quad \forall 1 \leq k \leq n, \forall 1 \leq j \leq |s_k| && (7) \\
x_{ijk} + x_{i'j'k} & \leq 1, \quad \forall 1 \leq k \leq n, \forall j' > j, \forall i' < i && (8) \\
\sum_{i \in S} \sum_{j=1}^{|s_k|} x_{ijk} & \leq |s_k| - 1, \quad \forall 1 \leq k \leq n, \forall S \text{ non-contiguous } \subset \{1, \ldots, N\} \text{ with } |S| = |s_k| && (9) \\
x_{ijk} + x_{i'j'k} & \leq 1, \quad \forall 1 \leq k \leq n, \forall j' > j, \forall i' < i && (10) \\
n y_i & \geq \sum_{k=1}^{n} \sum_{j=1}^{|s_k|} x_{ijk} \quad \forall i && (11) \\
x_{ijk} & \in \{0,1\} \quad \forall i, j, k \\
y_i & \in \{0,1\} \quad \forall i
\end{aligned}
$$

The constraint (7) says that each character in each s_k must be covered at least once. The constraint (8) ensures that the order in which characters are covered is increasing. The constraint (9) guarantees that no non-contiguous cover exists by restricting the number of characters covered by a non-contiguous subset of size $|s_k|$ to be at most $|s_k| - 1$ for each k. The constraint (10) makes sure that covers are consistent, so if the j^{th} character of s_k and the j^{th} character of $s_{k'}$ differ, then they cannot be covered by the same position i. The constraint (11) says that position i of the output string might cover a character of n input strings.