CSCI 2510 - Problem Set 5

Shay Mozes
shay@cs.brown.edu

1 Weighted Set Covet

Input: universe U, a collection S of k subsets $S_i \subseteq U$, positive cost c_i for each subset s_i.
Feasible output: a subcollection of S that covers all elements of U
Objective: minimize cost of feasible solution.

$$\begin{array}{rcl} \min & \sum_{i=1}^{k} c_i x_i & \text{s.t.} \\ \forall u \in U & \sum_{i=1}^{k} A_{ui} x_i \geq 1 & \\ \forall i & x_i \in \{0,1\} & \end{array}$$

Where

$$A_{ui} = \begin{cases} 1 & \text{if } u \in S_i \\ 0 & \text{otherwise} \end{cases}$$

x_i specifies whether subset S_i is in the solution, and we require that each element appears in at least one of the subsets.

2 Minimum Spanning Tree

Input: undirected graph $G = (V, E)$, edge costs c_e for each $e \in E$.
Feasible output: a spanning tree of G.
Objective: minimize cost of solution.
Note: see alternative formulation in the minimum Steiner tree problem.

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

x_e indicates whether edge e is in the spanning tree or not. The first constraint ensures we have at least $|V| - 1$ edges in the solution (any MST has exactly $|V| - 1$ edges). The second type of constraints makes sure there are no cycles 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 this applies to all subsets, the solution contains no cycles and hence it is a forest. Since it contains at least $|V| - 1$ edges, it only contains a single connected component, so the forest is a tree.

3 Minimum Steiner Tree

Input: undirected graph $G = (V, E)$, non-negative edge costs c_e for each $e \in E$, a subset $S \subseteq V$
Feasible output: a tree in G that spans all nodes in S.
Objective: minimize cost of output

$$\begin{array}{rcl} \min & \sum_{e \in E} c_e x_e & \text{s.t.} \\ \forall T \subseteq V : T \not\supseteq S & \sum_{e=(u,v): u \in T, v \in V \setminus T} x_e \geq 1 & \\ \forall e \in E & x_e \in \{0, 1\} & \end{array}$$

x_e indicates whether edge e is in the Steiner tree or not. The constraint ensures that all nodes of S are connected; For each cut of V that induces a cut of S, there must be at least one edge crossing that cut in the solution. We do not have to worry about cycles because this is a minimization problem, and any solution that contains a cycle is not minimal. This formulation can be used for MST with $S = V$.

4 Shortest s-t Path

Input: graph $G = (V, E)$, edge costs c_e for each $e \in E$, source and target nodes s, t.
Feasible output: a path connecting s to t in G.
Objective: minimize cost of path

$$\begin{array}{rcl} \min & \sum_{e \in E} c_e x_e & \text{s.t.} \\ \forall (s,t)\text{-cut } C & \sum_{e=(u,v): u \in C, v \in V \setminus C} x_e \geq 1 & \\ \forall e \in E & x_e \in \{0, 1\} & \end{array}$$

x_e indicates whether edge e is in the path or not. The constraint ensures that the set of chosen edges crosses any (s,t)-cut in G, so s and t are connected. The minimization ensures that we get a (minimal) simple path. This formulation works for both directed and undirected graphs.

5 Traveling Salesman Tour

Input: complete undirected graph $G = (V, E)$, non-negative edge costs c_e for each $e \in E$.
Feasible output: a simple cycle visiting all nodes in V.
Objective: minimize cost of tour

$$\begin{array}{rcl} \min & \sum_{e \in E} c_e x_e & \text{s.t.} \\ \forall v \in V & \sum_{e \text{ incident to } v} x_e \leq 2 & \\ \forall S \subseteq V & \sum_{e=(u,v): u \in C, v \in V \setminus C} x_e \geq 2 & \\ \forall e \in E & x_e \in \{0, 1\} & \end{array}$$

x_e indicates whether edge e is in the tour or not. The first constarint makes sure that the degree of each vertex is at most 2, so all cycles in the solution are simple. The second constraint make sure that the solution is 2-connected, so there must be a cycle that contains all nodes.

6 Multiway Cut

Input: undirected graph $G = (V, E)$, non-negative edge costs c_e for each $e \in E$, set of k terminals $\{s_1 \dots s_k\} \subseteq V$
Feasible output: a set of edges whose removal disconnects the terminals from each other.
Objective: minimize cost of solution

$$\begin{array}{rll} \min & \sum_{e\in E} c_e x_e & \text{s.t.} \\ \forall 1 \le i \le j \le k \quad \forall s_i\text{-to-}s_j \text{ path } p & \sum_{e\in p} x_e \ge 1 & \\ \forall e \in E & x_e \in \{0,1\} & \end{array}$$

x_e indicates whether edge e is removed or not. The constraints make sure that any path that connects a pair of terminals contains at least one edge which is removed.

7 Metric Minimum k-Center

Input: complete graph $G = (V, E)$, non-negative edge costs c_e satisfying the triangle inequality. a number k
Feasible output: a set of k nodes $s_1 \dots s_k$ in V.
Objective: minimize $max_v\{min_i c(v, s_i)\}$

$$\begin{array}{rcl} \min & d & \text{s.t.} \\ \forall u \in V & \sum_{v\in V} x_{uv} \ge 1 & \\ \forall v \in V & |V| y_v \ge \sum_u x_{uv} & \\ & \sum_v y_v \le k & \\ \forall u, v \in V \times V & d \ge x_{uv} c_{uv} & \\ \forall u, v \in V \times V & x_{uv} \in \{0,1\} & \\ \forall v \in V & y_v \in \{0,1\} & \\ & d \ge 0 & \end{array}$$

x_{uv} indicates whether node u is in node's v cluster. The first constraint says that every node is in some node's cluster. y_v indicates whether v is a center node or not. The second constraint makes sure that each node that appears asthe center for some node must have $y_v > 0$. The third constraint makes sure there are at most k centers. d is constrained to be greater than the distance of each node to it's assigned center by the fourth constraint.

8 Feedback Vertex Set

Input: undirected graph $G = (V, E)$, non-negative vertex costs c_v.
Feasible output: a set of nodes whose removal leaves an acyclic graph.
Objective: minimize cost of solution

$$\begin{array}{rll} \min & \sum_{v \in V} c_v x_v & \text{s.t.} \\ \forall \text{cycle} c \in G & \sum_{v \in C} x_v \geq 1 & \\ \forall e \in E & x_e \in \{0, 1\} & \end{array}$$

x_v indicates whether node v is in the feedback vertex set. The constraint makes sure that at least one vertex in every cycle in the original graph is removed.

9 Shortest Superstring

Input: a set of n strings $\{s_1 \dots s_n\}$over a finite alphabet Σ.
Feasible output: a string t that contains each s_i as a substring.
Objective: minimize the length of t.
Let N be the total sum of all strings $N = \sum |s_i|$. Let s_{kj} denote the j^{th} character of s_k.

$$\begin{array}{rc} \min & \sum y_i \\ \forall k \quad \forall j \leq |s_k| & \sum_{i=1}^{N} x_{ijk} \geq 1 \\ \forall k \quad \forall j' > j \quad \forall i' < i & x_{ijk} + x_{i'j'k} \leq 1 \\ \forall k \quad \forall S \text{ non-contiguous subset of} & \\ \{1 \dots N\} \text{ with } |S| = |s_k| & \sum_{i \in S} \sum_{j \leq |s_k|} x_{ijk} \leq |s_k| - 1 \\ \forall i \quad \forall k, k', u \leq |s_k|, v \leq |s_{k'}| \text{ with } s_{ku} \neq s_{k'v} & x_{iuk} + x_{ivk'} \leq 1 \\ \forall i & k y_i \geq \sum_k \sum_{j \leq |s_k|} x_{ijk} \\ \forall i, j, k & x_{ijk} \in \{0, 1\} \\ \forall i & y_i \in \{0, 1\} \end{array}$$

Clearly $|t| \leq N$. x_{ijk} indicates whether t_i covers the occurrence of s_{kj}. The first constraint says that each character in each s_k must be covered at least once. The second constraint ensures that the order in which characters are covered is increasing. The third constraint ensures 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 fourth constraint 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 y_is, defined by the fifth constraint, indicate whether a specific location in the output string is used to cover some character. The objective is to minimize the number of characters used in the cover. Since characters not used in the cover may be omitted, it is easy to construct a superstring from the values of the x_{ijk}s.