18.415/6.854 Advanced Algorithms	October 1, 2001

Problem Set 2

Due October 10, 2001

1. The Min s-t-Cut problem is the following:

   Given an undirected graph $G = (V, E)$, a weight function $w : E \to \mathbb{R}^+$, and two vertices $s, t \in V$, find

   $$Min\ s - t - Cut(G) = \min\{w(\delta(S)) : S \subset V, s \in S, t \notin S\}.$$

   where $\delta(S)$ denotes the cut

   $$\delta(S) = \{(i, j) \in E : |\{i, j\} \cap S| = 1\}$$

   and

   $$w(\delta(S)) = \sum_{e \in \delta(S)} w(e).$$

   (a) Argue (in just a few lines) that there is a polynomial-time algorithm to find a Min $s - t$-cut based on linear programming (remember Problem Set 1). (Be careful; problem set 1 defined the Min $s - t$-cut problem for a directed graph, while this problem considers undirected graphs.) [We will see a much more efficient algorithm for it (not based on linear programming) later this semester.]

   We are going to develop an algorithm for a generalization of the problem:

   Given an undirected graph $G = (V, E)$, $w : E \to \mathbb{R}^+$, and an even cardinality subset of vertices $T \subseteq V$, find

   $$Min\ T - Odd - Cut(G) = \min\{w(\delta(S)) : S \subset V, |S \cap T| = odd\}$$

   That is, we want to optimize over all cuts that separate T into two parts of odd size (since $|T|$ is even, $|S \cap T|$ odd implies that $T \setminus S|$ odd as well).

   (b) Suppose that $|T| = 2$, say $T = \{s, t\}$. What is the Min T-Odd-Cut then?

   (c) For a given $T \subseteq V$, call a cut $\delta(S)$ T-splitting if $\emptyset \neq S \cap T \neq T$.
   Using a s-t-Min-Cut algorithm, show how we can find the minimum T-splitting cut in polynomial time. Can you do it in at most $|T|$ calls to a Min s-t-Cut algorithm?

(d) For any two sets C and D ($\emptyset \neq C, D \subset V$), prove the inequality that

$$w(\delta(C \setminus D)) + w(\delta(D \setminus C)) \leq w(\delta(C)) + w(\delta(D)).$$

(e) Prove that if $\delta(C)$ is a minimum T-splitting cut then there is a minimum T-odd-cut $\delta(D)$ such that either $D \subseteq C$ or $C \subseteq D$.

Hint: Use the inequality proved above.

(f) Use the previous observation to design a recursive algorithm which solves Min T-Odd-Cut in polynomial time. (Hint: possibly think about modifying the graph.) How many calls (in $O(\cdot)$ notation) to a Min s-t-Cut algorithm does your algorithm perform?

2. Use the ellipsoid method to solve the minimum weight perfect matching problem (there is a more efficient combinatorial algorithm for it, but here we will use the power of the ellipsoid algorithm):

Given an undirected graph $G = (V, E)$ and a weight function $w : E \to \mathbb{N}$, find a set of edges M covering every vertex exactly once (a perfect matching) with the minimum total weight.

In order to formulate this problem as a linear program, we define the V-join polytope:

$$P = conv\{\chi_M \in \{0,1\}^E : M \text{ is a perfect matching}\}$$

where χ_M is the characteristic vector of M ($\chi_M(e) = 1$ if $e \in M$ and 0 otherwise). The convex hull $conv(A)$ is defined as $\{\sum_i \lambda_i x_i : x_i \in A, \lambda_i \geq 0, \sum_i \lambda_i = 1\}$ (where the summation is finite).

(a) Argue that the vertices of P are the characteristic vectors of perfect matchings. Deduce that if we can optimize $\sum_e w_e x_e$ over P, we would find a minimum weight perfect matching.

(b) Suppose now that we can decide (via linear programming or some other way) whether $P \cap \{x : w^T x \leq \lambda\}$ is empty or not, for any given λ (remember all weights w_e are integers). Show that by calling an algorithm for this decision problem a polynomial number of times (in the size of the input, i.e. $|V|$, $|E|$ and $\log(w_{max})$), we can find the weight of the minimum-weight perfect matching.

(c) With the same assumptions as in the previous part, can you also find a minimum-weight perfect matching (not just its weight, but also which edges are in it) in polynomial time? (There might be several perfect matching having the same minimum weight, but here you need to produce only one of them. Also, the algorithm does not need to be extremely efficient, just polynomial.)

Due to Jack Edmonds, the perfect matching polytope can be described by the following inequalities:

- $\forall e \in E; x_e \geq 0$
- $\forall v \in V; \sum_{e \in \delta(\{v\})} x_e = 1$
- $\forall W \in V, |W| = \text{odd}; \sum_{e \in \delta(W)} x_e \geq 1$

(d) Show that every vector in P satisfies the above inequalities.

Take the other implication for granted (every vector satisfying these inequalities is in P).

(e) How many inequalities do we have in this complete description of P? Can we just use any polynomial-time algorithm for linear programming to optimize over P?

(f) Show how we can use the ellipsoid method to decide if there exists a perfect matching of weight at most λ in polynomial time. How would you select the initial ellipsoid? How would you take care of the equality constraints in the description of P? When can you stop?