Exam “Discrete Optimization”

Monday, December 19, 2016, 13:30 – 16:30

- Answers have to be in English.
- Use of calculators, mobile phones, and other electronic devices is not allowed.
- The exam consists of six problems. Please start a new page for every problem.
- The total number of points is 50.

1. Traveling Salesman Problem

Our goal is to find an approximation algorithm for **MaxTSP**:

Instance: undirected, complete graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}^+$.

Solution: a Hamiltonian cycle $H \subseteq E$ of G.

Goal: **maximize** $w(H) = \sum_{e \in H} w(e)$.

For an instance $G = (V, E)$ and w of MaxTSP, let $H^\star$ be a Hamiltonian cycle of G of maximum weight.

To approximate MaxTSP, we use the following algorithm:

$M = \emptyset$
while $E \neq \emptyset$ **do**
    choose the heaviest edge $e = \{u, v\} \in E$, breaking ties arbitrarily
    $M = M \cup \{e\}$
    remove all edges incident to u or v from E (thus, in particular, we remove e)
end while
connect the edges in M in an arbitrary way to obtain a Hamiltonian cycle H

In the following, we assume that the number n of nodes is even.

(a) *(2 points)* Let $M^\star$ be a maximum-weight perfect matching of the graph G with edge weights w.

Prove that $w(M) \geq \frac{1}{2} \cdot w(M^\star)$.

(b) *(2 points)* Prove that $w(M^\star) \geq \frac{1}{2} \cdot w(H^\star)$.

(c) *(2 points)* Prove that the algorithm given above is a 4-approximation for MaxTSP and has a running-time of $O(n^2 \log n)$.

2. Shortest Path Trees and Minimum Spanning Trees

(6 points) Let $G = (V, E)$ be a connected, undirected graph with non-negative edge weights w. For $v \in V$, let S_v denote a shortest path tree rooted at v, i.e., S_v contains shortest paths from v to all other nodes in G. Note that S_v is not necessarily unique. If S_v is not unique, you are not allowed to choose which tree you get. This means that the statement in the following must hold for all possible choices of S_v for $v \in V$.

Prove the following statement: There exists a minimum-weight spanning tree T of G with

$$T \subseteq \bigcup_{v \in V} S_v.$$

(Here, we consider T and S_v for $v \in V$ as sets of edges.)

3. Min-Cost Flows

(10 points) For a flow network $G = (V, E)$ with edge capacities $u = (u_e)_{e \in E}$ and balances $b = (b_v)_{v \in V}$ and a feasible flow f, let $H_f = (V, F_f)$ be the following undirected graph:

$$F_f = \big\{\{u, v\} \mid \text{both } (u, v) \text{ and } (v, u) \text{ are contained in } G_f\big\}.$$

Here, G_f denotes the residual network with flow f. This means that F_f contains an undirected edge between nodes u and v if and only if both (u, v) are (v, u) are present in G_f.

We call a flow network $G = (V, E)$ 2-cycle-free if there does not nodes $u, v \in V$ with $(u, v), (v, u) \in E$.

Prove that the following two statements are equivalent for all 2-cycle-free flow networks $G = (V, E)$ with edge capacities $u = (u_e)_{e \in E}$ and balances $b = (b_v)_{v \in V}$ and feasible flows f of this network:

(I) There exist edge costs $c : E \to \mathbb{R}$ such that f is the *unique* minimum-cost flow with respect to c in this network.

(II) H_f is a forest.

4. NP-Completeness

BCSP (short for "bicriteria shortest path") denotes the following optimization problem:

Instance: directed graph $G = (V, E)$, nodes $s, t \in V$, costs $c : E \to \mathbb{N}$, lengths $\ell : E \to \mathbb{N}$, cost budget $C \in \mathbb{N}$.

Solution: s-t path Q with $c(Q) = \sum_{e \in Q} c(e) \leq C$.

Goal: minimize length $\ell(Q) = \sum_{e \in Q} \ell(e)$.

The decision version of BCSP is the following problem: Given an instance of BCSP and a number $L \in \mathbb{N}$, does there exist a solution Q with $\ell(Q) \leq L$?

(a) *(7 points)* Prove that BCSP is NP-hard. You do not have to prove that the decision version of BCSP is in NP.

Hint: Knapsack is the following NP-hard problem:

Instance: weights $w_1, \ldots, w_n \in \mathbb{N}$, profits $p_1, \ldots, p_n$, bound $W \in \mathbb{N}$.

Solution: $U \subseteq \{1, \ldots, n\}$ with $w(U) = \sum_{i \in U} w_i \leq W$.

Goal: maximize $p(U) = \sum_{i \in U} p_i$.

The decision version of Knapsack is the following problem: Given an instance of knapsack and a $P \in \mathbb{N}$, does there exist a feasible solution U with profit $p(U) \geq P$?

(b) *(7 points)* Devise an algorithm that solves BCSP and whose running-time is bounded by a polynomial in C, the number n of vertices of G and the number m of edges of G. It suffices if your algorithm outputs the length of a shortest path of costs at most C – it is not necessary to output the path itself.

You do not have to give a proof of correctness, but you have to explain briefly why your algorithm works and why it has the running-time that you claim.

5. 1-Trees

(5 points) Let $G = (V, E)$ be an undirected graph with non-negative edge weights w. A 1-tree of G is a connected subgraph L that has $|V| = n$ edges. (The name 1-tree comes from the fact that L consists of a spanning tree plus one additional edge.)

Consider the following algorithm:

compute a minimum spanning tree of G with respect to w
let e be an edge of minimum weight among all edges in $E \setminus T$
$L = T \cup \{e\}$

Prove that this algorithm computes a 1-tree of minimum weight.

6. Miscellaneous Questions

Are the following statements true or false? Justify your answer. This justification can be a short proof, a reference to a theorem of the lecture, a counterexample, ...

(a) *(2 points)* For all directed graphs $G = (V, E)$ with non-negative edge weights w, the following holds: If there is a unique edge $e^\star \in E$ of minimum weight, i.e., $w_{e^\star} = \min\{w_e \mid e \in E\}$, then for every $v \in V$, there exists a shortest path tree rooted at v that contains $e^\star$.

(b) *(2 points)* If there is a pseudo-polynomial algorithm for TSP, then there is a polynomial-time algorithm for Knapsack.

(c) *(2 points)* Let $(S, \mathcal{I})$ be an independent set system. This means that S is a finite set and $\mathcal{I} \subseteq \mathcal{P}(S)$ satisfies (i) $\emptyset \in \mathcal{I}$ and (ii) for all X and Y with $Y \subseteq X \in \mathcal{I}$, we have $Y \in \mathcal{I}$.

If $(S, \mathcal{I})$ is not a matroid, then there exist weights for the elements in S such that the greedy algorithm does not compute an independent set of maximum weight.

(d) *(3 points)* For all undirected, complete graphs $G = (V, E)$ with edge weights w, the following holds: There exists a number t such that for all minimum spanning trees T of G, we have

$$\max\{w_e \mid e \in T\} = t.$$