Technische Universität München
Fakultät für Mathematik
Dr. Jannik Matuschke

Network flows
Summer term 2016

Problem set 2

Discussion: May 12, 2016

Exercise 2.1
The *transportation problem* is the special case of the minimum cost flow problem, where D is a bipartite graph and all capacities are infinite. Show that any instance of minimum cost flow problem on a network $D = (V, A)$ can be transformed into an equivalent instance of the transportation problem on a network $D' = (V', A')$ with $|V'| = |V| + |A|$ and $|A'| = 2|A|$.

Exercise 2.2
Let $D = (V, A)$ be a digraph, let $u \in \mathbb{R}^A_+$, let $c \in \mathbb{R}^A$ and let $b, b' \in \mathbb{R}^V$. Let x be a b-flow in D for capacities u and let x' be a b'-flow in D_x (the residual network of x) for capacities u_x. Extend x' by defining $x'(a) := 0$ for all $a \in \overleftrightarrow{A} \setminus D_x$. For all $a \in A$, define $x''(a) := x(a) + x'(a) - x'(\overleftarrow{a})$.

a) Show that x'' is a $(b + b')$-flow in D for capacities u.

b) Argue that a) implies the following two results from the lecture: (1) Augmenting an s-t-flow x along an x-augmenting path P by $\gamma := \min_{a \in P} u_x(a)$ yields an s-t-flow of value $\mathrm{val}(x) + \gamma$. (2) Augmenting a b-flow x along an x-augmenting cycle C by $\gamma := \min_{a \in C} u_x(a)$ yields a b-flow of cost $\mathrm{cost}(x) + \gamma \sum_{a \in C} x(a)$.

Exercise 2.3
Remeber the generalization proposed by Prof. Wright in Problem 1.6. After seeing how you disproved his claim, he proposes to look at the following special case: Assume there is $R \in \mathbb{R}_+$ and $r' \in \mathbb{R}^A_+$ such that $r(P) = R - \sum_{a \in P} r'(a)$ for all $P \in \mathcal{P}$. Show that for this special case Claims a) and b) stated in Problem 1.6 are true (for Claim b) you may assume that R and r' are given as part of the input).

Exercise 2.4
In order for the minimum mean cost cycle canceling algorithm to work, we need to be able to find such a cycle efficiently. The following insight turns out to be very helpful for this.
Let $D = (V, A)$ be a digraph with $n := |V|$ and $c \in \mathbb{R}^A_+$ and assume there is a node $s \in V$ be such that each node $v \in V$ is reachable from s. For $v \in V$ and $k \in \mathbb{Z}_+$ define

$$F_k(v) := \min \left\{ \sum_{i=1}^{k} c(a_i) \,:\, (a_1, \ldots, a_k) \text{ is an } s\text{-}v\text{-walk in } D \right\}$$

as the minimum cost of an s-v-walk of length exactly k in D.
Let $\mu(D, c) := \min \left\{ \frac{\sum_{a \in C} c(a)}{|C|} \,:\, C \text{ is a cycle in } D \right\}$ be the minimum mean cycle cost in D.

a) Show that if $\mu(D, c) = 0$ then $\min_{v \in V} \max_{0 \le k \le n-1} \frac{F_n(v) - F_k(v)}{n-k} = 0$.

b) Use a) to show that $\mu(D, c) = \min\limits_{v \in V} \max\limits_{0 \le k \le n-1} \dfrac{F_n(v) - F_k(v)}{n - k}$ is always true.

Remark: An s-v-walk in D is a sequence of arcs $(a_1, \ldots, a_k)$ such that $a_i \in A$ for all $i \in \{1, \ldots, k\}$, $\mathrm{tail}(a_1) = s$, $\mathrm{head}(a_k) = v$ and $\mathrm{head}(a_i) = \mathrm{tail}(a_{i+1})$ for $i \in \{1, \ldots, k-1\}$. Note that this definition allows the walk to use arcs multiple times. Furthermore, we introduce the convention $\min \emptyset := \infty$.

Exercise 2.5
A *loop* is an arc such that $\text{head}(a) = \text{tail}(a)$. Two arcs a, a' are *parallel* if $\text{head}(a) = \text{head}(a')$ and $\text{tail}(a) = \text{tail}(a')$. A digraph is *simple* if it does not contain any loops and pairs of parallel arcs.

a) Show that if $D = (V, A)$ is a simple digraph such that there is a node $s \in V$ such that every $v \in V$ is reachable from s, then $|V| - 1 \leq |A| \leq |V| \cdot (|V| - 1)$.

b) You bought a software from a shady algorithms dealer, who claimed that it solves the maximum flow problem very quickly. At home, you discover that, while the software is indeed very fast and computes a maximum flow, it only accepts as input instances of the maximum flow problem for which the underlying digraph is simple. What can you do to solve instances where this is not the case?

c) You discover that the software also can solve the minimum cost flow problem, again with the limitation that it only works on simple graphs. Can you apply the same trick as in b)? Can you do something else?