MTAT.03.286: Advanced Methods in Algorithms **University of Tartu**

Final exam

Course staff: Vitaly Skachek, Pille Pullonen May 25th, 2016

Student name: _____

Student ID: _____

1. This exam contains 10 pages. Check that no pages are missing.
2. It is possible to collect up to 110 points. Try to collect as many points as possible.
3. Justify and prove all your answers (where applicable).
4. All facts and results that were proved or stated in the class can be used in your solution without a proof. Such results need to be rigorously formulated.
5. Any printed and written material is allowed in the class. No electronic devices are allowed.
6. Exam duration is 3 hours.
7. Good luck!

Question 1	
Question 2	
Question 3	
Question 4	
Total	

Question 1 (25 points). You are given the following linear-programming (LP) problem:

$$\begin{array}{rrcrcrcl} \textbf{max} & x_1 & - & 3x_2 & + & x_3 & & \\ \textbf{s.t.} & x_1 & + & x_2 & & & \leq & 1 \\ & x_1 & + & & & x_3 & \leq & 2 \\ & & & x_2 & + & 2x_3 & \leq & 2 \\ & x_1 \geq 0, & & x_2 \geq 0, & & x_3 \geq 0 & & \end{array}$$

(a) Solve this LP problem by using the simplex method.

(b) Formulate the dual LP problem.

(c) Verify that your solution of (a) is optimal by the substitution of the point $(0, 1, 0)$ into the dual problem. Explain your answer.

Solution

(a) Steps of the simplex algorithm

The simplex algorithm uses the following two rules:

1. Choose a column (variable) with a positive coefficient in the last row. If no such column exists then the current solution is optimal.
2. For each row i where this column c has a positive entry, compute the ratio $\frac{b[i]}{c[i]}$ between this entry and the right hand side of the corresponding constraint. Choose the row with the smallest ratio.

In the first iteration we choose the first column and first row

$$\left(\begin{array}{ccc|ccc|c} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & -1 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 0 & 1 & 2 \\ \hline 0 & -4 & 1 & -1 & 0 & 0 & -1 \end{array}\right).$$

At the second iteration we choose third column and second row:

$$\left(\begin{array}{ccc|ccc|c} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & -1 & 1 & 0 & 1 \\ 0 & 3 & 0 & 2 & -2 & 1 & 0 \\ \hline 0 & -3 & 0 & 0 & -1 & 0 & -2 \end{array}\right).$$

We obtain that the optimal solution value is 2 and this is obtained for $x_1 = 1$, $x_2 = 0$, $x_3 = 1$.

(b) Dual

$$
\begin{array}{ll}
\textbf{min} & y_1 + 2y_2 + 2y_3 \\
\textbf{s.t.} & y_1 \ + \ y_2 \qquad\quad \geq \ 1 \\
& y_1 \qquad + \ y_3 \geq \ -3 \\
& \qquad\quad y_2 \ + 2y_3 \ \geq \ 1 \\
& x_1 \geq 0, \ x_2 \geq 0, \ x_3 \geq 0
\end{array}
$$

(c) Checking the point

The given point $(0, 1, 0)$ is a feasible point of the dual problem and the value of the dual objective function on this point is 2. From the strong duality theorem we conclude that this is an optimal solution for the dual problem. We obtain a confirmation that the optimal value 2 is computed correctly for the first part of the problem.

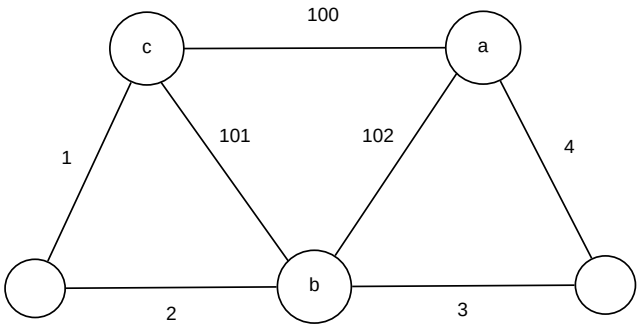
Question 2 (25 points). Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a finite undirected graph with the real-valued weight function w defined on the edges, such that $w(e) > 0$ for all edges $e \in \mathcal{E}$. It is known that the weights of edges in $\mathcal{G}$ are all different. Let e_1 be some edge, and $\mathcal{C}$ be a simple circuit that contains e_1. Prove or disprove the following statements.

(a) If e_1 has the lowest weight in $\mathcal{C}$, then there exists a minimum spanning tree in $\mathcal{G}$ that contains e_1.

(b) If e_1 has the highest weight in $\mathcal{C}$, then there exists no minimum spanning tree in $\mathcal{G}$ that contains e_1.

(a) The statement is wrong

We show that by a counterexample. Consider a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ as depicted below. The edge weights are shown in the figure.



Let e_1 be an edge $\{a, c\}$, and $\mathcal{C}$ be a simple circuit a — b — c — a. The edge e_1 has the lowest weight among the edges in $\mathcal{C}$. However, there is only one minimum spanning tree in $\mathcal{G}$, which contains the edges of weights 1, 2, 3 and 4. This minimum spanning tree does not contain e_1.

(b) The statement is correct

By contrary, assume that there exists a minimum spanning tree $T = (\mathcal{V}, \mathcal{E}_T)$ of $\mathcal{G}$ that contains $e_1 = \{u, v\}$. If we remove e_1 from T, the remaining graph is not connected, and it consists of two connected components. Denote these connected components by C_1 and C_2. Without loss of generality assume that $u \in C_1$ and $v \in C_2$ (otherwise we exchange the names of C_1 and C_2).

There exists an edge $e_2 = \{a, b\} \in \mathcal{C}$, $e_2 \neq e_1$, such that one of its points is in C_1 and the other endpoint is in C_2 (since $u \in C_1$ and $v \in C_2$, and there is a path in $\mathcal{C} \backslash \{e_1\}$ that connects u and v, there should be at least one edge in this path that connects a vertex in C_1 with a vertex in C_2.)

Consider a graph H consisting of the vertices $\mathcal{V}$ and the edges $\mathcal{E}_T \cup \{e_2\} \backslash \{e_1\}$. The graph H is connected:

- any two vertices in C_1 are connected by a path since C_1 is a connected component;
- any two vertices in C_2 are connected by a path since C_2 is a connected component;

- any vertex a_1 in C_1 is connected with any vertex b_1 in C_2 by a path, since the vertices $a_1, a \in C_1$ are connected, vertices $b, b_1 \in C_2$ are connected, and $e_2 = \{a, b\}$ is an edge.

The graph H is connected, it has the same number of vertices and edges as T. Therefore, H is a tree. Since H contains all the vertices of $\mathcal{V}$, it is a spanning tree. Since e_1 is the highest weight edge in $\mathcal{C}$, the tree H has weight smaller than that of T. Contradiction to minimality of T.

Question 3 (25 points). You are given an integer number k and a flow network $\mathcal{N}(\mathcal{G}(\mathcal{V}, \mathcal{E}), s, t, c)$, such that all edge capacities are $c(e) = 1$. Propose an algorithm, which finds a subset of k edges of $\mathcal{E}$, such that by deleting these edges from $\mathcal{G}$, the maximum flow from s to t in the resulting network is minimal. The algorithm should have time complexity of $O(|\mathcal{V}|^2|\mathcal{E}|)$. Prove correctness of that algorithm and analyze its time complexity.

Solution

The core idea is to find a minimum cut $c(S : \overline{S})$ in the graph and then to remove k edges that belong to this cut. This decreases the flow in the resulting graph by k based on the max-flow min-cut theorem. In case there are less than k edges in the minimum cut, we can make the graph disconnected by removing all the edges in this cut and a suitable number of randomly chosen other edges.

In a full solution, one should prove that removing the edges from the minimum cut indeed gives the least possible flow F in the new network. This is trivial in case we make the graph disconnected. For the case the network remains connected, assume, by contradiction that there exists some other set of k edges that can be removed to obtain a smaller flow $F' < F$. Let F'' be the flow in the initial network, and then it holds $F = F'' - k$.

Consider the minimum cut that corresponds to the flow F'. We can add at most k edges to obtain a cut in the initial network. Consider the respective cut in the initial network before removing the edges, this cut has capacity $\leq F' + k < F + k = F''$. This is a contradiction: as the maximum flow in the initial network was F'', the minimum cut has to have the same capacity too.

The complexity of the algorithm is that of finding the maximum flow and then finding minimum cut. You can do this by using Dinitz algorithm to find the flow and then setting the cut to be all such edges that can be reached from the source vertex s using augmented paths. In total, the time complexity of the algorithm is equal to that of the Dinitz algorithm, $O(|\mathcal{V}|^2 \cdot |\mathcal{E}|)$. (By observing that the edges have capacities 0 and 1, even better complexity can be argued.)

Question 4 (35 points).

A student is moving to a new apartment. He needs to take n objects a_1, a_2, ..., a_n with him. The object a_i has weight $w_i > 0$ kilograms, $i = 1, 2, \cdots, n$. The student's car can carry at most k kilograms at a time (assume that there are no other restrictions on the objects placed in the car). The student wants to minimize m, which is the number of times he drives from the old apartment to the new apartment.

Consider the following greedy algorithm. The objects are ordered in an arbitrary order. The student loads the objects into the car in that order as long as the total weight of the loaded objects does not exceed k. If the next object a_i does not fit into the car (the load exceeds k kilograms), then the student leaves a_i in the old apartment, and drives with the items that are already loaded. The student then returns and continues the same algorithm starting with the object a_i.

Denote by ℓ_j, $j = 1, 2, \cdots, m$, the load of the car when the student drives j-th time from the old apartment to the new apartment.

(a) Prove that for any $j = 1, 2, \cdots, m-1$ it holds that $\ell_j + \ell_{j+1} > k$.

(b) Show that the above greedy algorithm achieves approximation factor 2.

(c) Show that for every set of objects there exists an initial ordering such that m is optimal.

(d) Show an example of k, of n, of weights $w_1, w_2, \cdots, w_n$, and of the object ordering, such that by using the above algorithm, $m \geq 1.9 \cdot \text{OPT}$, where OPT is the optimum value.

(a) Weight of objects in two consecutive rides

Assume without loss of generality that the objects are ordered in the following order: a_1, a_2, ..., a_n (otherwise, we could rename them).

By contrary, assume that $\ell_j + \ell_{j+1} \leq k$. Let w_i be the weight of a_i, which is the first object in the load ℓ_{j+1}. Then,

$$\ell_j + w_i \leq \ell_j + \ell_{j+1} \leq k \, .$$

Then, according to the algorithm, the student should have added the object a_i to the load ℓ_j. Contradiction. It follows that $\ell_j + \ell_{j+1} > k$.

(b) Approximation factor

Denote $W = \sum_{i=1}^{n} w_i$ the total weight of all objects.

In the optimal solution, each time the car carries at most k kilograms. Therefore, the optimal number of rides OPT is

$$\text{OPT} \cdot k \geq W \, . \tag{1}$$

In the solution given by the algorithm that the student uses, from (a), the total weigh in two consecutive trips exceeds k kilograms (except for the last ride). Therefore, the average weight per

trip (except for the last trip, if the number of trips is odd) is larger than $k/2$. Therefore, the number of car rides, m, satisfies

$$m \cdot \frac{k}{2} < W \text{ if } m \text{ is even (2 trips required for each } k \text{ kilograms)} ,$$

and

$$(m-1) \cdot \frac{k}{2} < W \text{ if } m \text{ is odd (2 trips required for each } k \text{ kilograms + something is left)} .$$

From (1) and (2), in both cases we can write

$$(m-1) \cdot \frac{k}{2} < W \leq \text{OPT} \cdot k . \tag{2}$$

After simplification we have

$$m - 1 < 2 \cdot \text{OPT} .$$

Since OPT and m are both integers, we obtain $m \leq 2 \cdot \text{OPT}$.

(c) Optimal solution

Assume, without loss of generality (otherwise rename the objects), that in the optimal solution, in trip i the objects $\{a_{1,i}, a_{2,i}, \cdots, a_{r_i,i}\}$ are moved together. Then, we can order all the objects as follows:

$$a_{1,1}, a_{2,1}, \cdots, a_{r_1,1}, \quad a_{1,2}, a_{2,2}, \cdots, a_{r_2,2}, \cdots \quad a_{1,m}, a_{2,m}, \cdots, a_{r_m,m} .$$

Claim: after i trips, the objects $a_{1,1}, a_{2,1}, \cdots, a_{r_1,1}, \cdots a_{1,i}, a_{2,i}, \cdots, a_{r_i,i}$ are moved, for $i = 1, 2, \cdots, m$.

Proof: by induction on i.

- For $i = 1$, it is obviously true.
- Assume that the statement is true for some i, and consider trip $i+1$. In trip $i+1$ we need to make sure that the objects $a_{1,1}, a_{2,1}, \cdots, a_{r_1,1}, \cdots a_{1,i+1}, a_{2,i+1}, \cdots, a_{r_i,i+1}$ are moved. However, by inductive assumption, in trip i, the following objects are moved: $a_{1,1}, a_{2,1}, \cdots, a_{r_1,1}, \cdots a_{1,i}, a_{2,i}, \cdots, a_{r_i,i}$ (and maybe some additional objects too). Therefore, the additional objects that we need to move in trip $i+1$ are a subset of $a_{1,i+1}, a_{2,i+1}, \cdots, a_{r_i,i+1}$ and their weight is smaller or equal to k. So they fit into the $i+1$-st car.

(d) Factor ≥ 1.9

Assume that $k = 20$ kilograms, and there are 40 objects, whose weights are (according to the ordering of the objects):

$$20, 1, 20, 1, 20, 1, \cdots, 20, 1 .$$

By using the above algorithm, the student will have to make $m = 40$ trips. On the other hand, in the optimal solution, all 1 kilogram objects are moved in the same trip. Then, OPT= 21. We have

$$\frac{m}{\text{OPT}} = \frac{40}{21} > 1.9 .$$