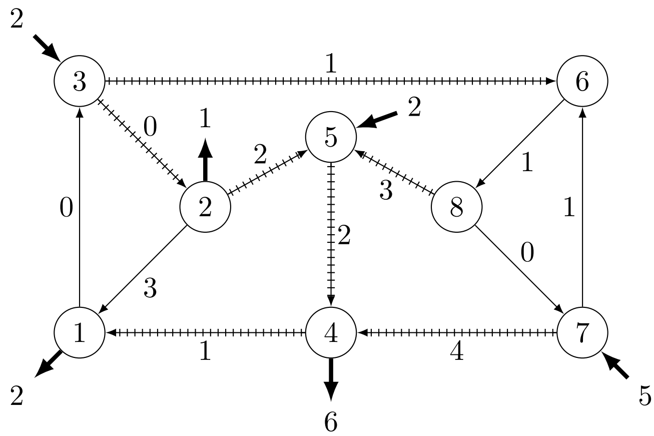
**Mathematics of Operational Research** **Example Sheet 2**
**R. Weber** **November 12, 2015**

**1**. Use the network simplex method to find a minimum cost flow of the following uncapacitated flow problem, where each edge is labeled with its cost. Start from the spanning tree indicated by hatched edges. If at some point you should encounter a situation where no positive amount of flow can be pushed around a cycle, continue to apply the network simplex method by pushing an amount of zero and explain what happens. You will need to be careful as you make your second and third pivots since this example has some degeneracy.

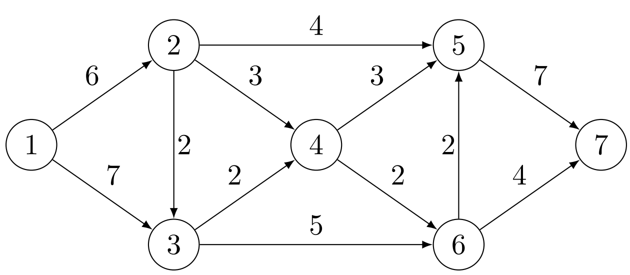



**2**. Prove or disprove that for a Hitchcock transportation problem with strictly positive costs, the optimal cost is weakly increasing in supply and demand as long as the problem remains feasible, i.e. the optimal cost cannot decrease as the supply and demand of some vertices increases.

**3**. Use the network simplex method to solve the transportation problem described by the following tableau:

| | | | |
|---|---|---|---|
| 7 | 4 | 9 | 20 |
| 8 | 12 | 5 | 42 |
| 3 | 11 | 7 | 18 |
| 39 | 34 | 7 | |

**4**. Consider the following network, where the number on each edge indicates its capacity:




Find the maximum flow from vertex 1 to vertex 7, and prove that this flow is indeed optimal.

**5**. Suppose that the pieces of 5 identical 200-piece jigsaw puzzles are scrambled and grouped into 200 piles of 5 pieces each. Prove or disprove that it is possible to pick exactly one piece from each of the piles and assemble these pieces into a complete puzzle.

**6**. Consider $n$ boys and $n$ girls where each knows exactly $k$ of the opposite gender. Is it possible to make a dance scheme of $k$ rounds in which each child will dance exactly once with the each of the other gender they know?

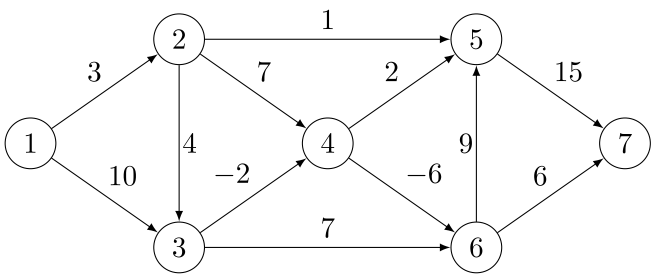
**7**. Consider a directed graph and two distinct vertices $s$ and $t$. Prove that the maximum number of vertex disjoint paths from $s$ to $t$ equals the minimum number of vertices whose removal will disconnect $s$ from $t$. Deduce that in a bipartite graph the size of the minimum vertex cover equals the cardinality of a maximum matching.

**8**. Use the Hungarian algorithm to solve the minimum cost assignment problem described in the following tableau:

| | 0 | 0 | $-4$ | |
|---|---|---|---|---|
| 4 | 7 | **1** 4 | 9 | $S1$ |
| 5 | 8 | 5 | 12 | $S2$ |
| 3 | **1** 3 | 11 | 7 | $S3$ |
| | $C1$ | $C2$ | $C3$ | |

Some initial node numbers $4, 5, 3$ and $0, 0, -4$ are given, along with an initial assignment of supplier 1 to customer 2 and supplier 3 to customer 1. This is a maximal matching in the existing equality graph.

**9**. Consider the following network, where the number on each edge indicates its length:




(a) Use the Bellman-Ford algorithm to find a shortest path from vertex 1 to vertex 7.

(b) Explain how to find a shortest path between all pairs of vertices, using an approach that minimizes asymptotic running time. Use it to find shortest distances from $i$ to 5, for all $i$.

**10**. Describe a polynomial-time 2-approximation algorithm for instances of the TSP satisfying $c_{ij} = c_{ji}$ and $c_{ik} \leq c_{ij} + c_{jk}$ for all $i, j, k \in V$. Start from the observation that for an arbitrary tree, there exists a walk that visits every edge in the tree exactly twice and returns to the initial vertex. Then show that following this walk in a minimum-cost spanning tree, and skipping vertices already visited, yields a TSP tour with the desired properties.

Prove that there is no $\alpha$-approximation algorithm for the general TSP unless P=NP. You may assume the Hamiltonian cycle decision problem is NP-complete.

**11**. Show that the optimal value in the max-cut problem is at least $|E|/2$. Hint. Think about putting each vertex in $S$ independently with probability 1/2.

Consider an algorithm which constructs a cut $(S, V \setminus S)$ by taking the vertices in the order $j = 1, 2, \dots, |V|$, and adds $j$ to $S$ or $V \setminus S$ depending on which choice produces the greater number of edges of the form $\{i, j\}$, $i < j$, crossing between $S$ and $V \setminus S$. Show that this is a 1/2-approximation algorithm for the max-cut problem.

**12**. A subset of vertices $S$ of the graph $G = (V, E)$ is said to be independent if there is no edge between two vertices in $S$. The maximum size of an independent set is called the stability number of $G$, denoted $\alpha(G)$. Calculating it is NP-hard. Suppose $S$ is independent. Let $x \in \{0, 1\}^{|V|}$ be such that $x_i = 1$ or 0 as $i \in S$ or $\notin S$. Let $X = xx^T/|S|$. Show that $X$ is positive semidefinite symmetric and that $\mathrm{tr}(X) = 1$, and $X_{ij} = 0$ for all $(i, j) \in E$. Describe a semidefinite programming whose solution provides an upper bound on $\alpha(G)$.

**13**. Use Dakin's method to solve the following IP:

$$\begin{aligned} \text{maximize} \quad & 8x_1 + 5x_2 \\ \text{subject to} \quad & x_1 + x_2 \leq 6 \\ & 9x_1 + 5x_2 \leq 45 \\ & x_1, x_2 \geq 0, \ x_1, x_2 \in \mathbb{N} \end{aligned}$$

Instead of carrying out iterations of the simplex method, you may draw the feasible set in $\mathbb{R}^2$ and use this drawing to explain how Dakin's method proceeds.

**14**. Consider a network $(V, E)$ with $V = \{1, \dots, 6\}$, $E = V \times V$, and edge costs $c_{ij}$ for $(i, j) \in E$ given by

$$C = \begin{pmatrix} 0 & 6 & 7 & 6 & 7 & 6 \\ 6 & 0 & 5 & 6 & 6 & 6 \\ 7 & 5 & 0 & 5 & 7 & 6 \\ 6 & 6 & 5 & 0 & 8 & 9 \\ 7 & 6 & 7 & 8 & 0 & 5 \\ 6 & 6 & 6 & 9 & 5 & 0 \end{pmatrix}.$$

(a) Find a TSP tour by starting from vertex 1 and applying the nearest neighbor heuristic, breaking ties toward vertices with smaller index. Improve this tour as much as possible using local search with the 2-OPT neighborhood.

(b) For each $i \in V$, find a minimum cost spanning tree of the network obtained by removing vertex $i$ from $(V, E)$. Use the information thus obtained to derive a lower bound on the cost of any TSP tour, and conclude that the TSP tour found above is optimal.

(c) Describe a branch-and-bound method for the TSP that combines the above lower bound with an appropriate branching rule.

**15**. In the facility location problem $n$ facilities are to be assigned to $n$ locations. The rate of flow between facilities $i$ and $j$ is $a_{ij}$. The cost per unit flow between locations $k$ and $\ell$ is $c_{k\ell}$. It is desired to locate one facility per location and minimize running cost. This means choosing a permutation $\sigma$ to minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}c_{\sigma(i)\sigma(j)} = \mathrm{tr}(AXCX^T) \tag{1}$$

where $A$ and $C$ are symmetric matrices and $X \in \mathcal{P}$, the set of permutation matrices. By taking $X_{ij} = \delta_{j,\sigma(i)}$ verify that the left and right hand sides of (1) express the same thing.

Let $\mathcal{O}$ be the set of $n \times n$ orthogonal matrices, i.e. if $X \in \mathcal{O}$ then $XX^T = X^TX = I$. We know that one can write $A = PDP^T$ and $C = QEQ^T$, where $P, Q \in \mathcal{O}$, and $D, E$ are diagonal, having eigenvalues $\{\alpha_i\}$ and $\{\beta_j\}$, respectively. Show that

$$\min_{X\in\mathcal{P}} \mathrm{tr}(AXCX^T) \geq \min_{X\in\mathcal{O}} \mathrm{tr}(DXEX^T) = \min_{X\in\mathcal{O}} \sum_{i,j} \alpha_i\beta_j X_{ij}^2. \tag{2}$$

Verify that if $X \in \mathcal{O}$ then $Z = (X_{ij}^2)$ is doubly stochastic, i.e. $\sum_i Z_{ij} = 1$ and $\sum_j Z_{ij} = 1$. It can be shown that the set of doubly stochastic matrices is the convex hull of $\mathcal{P}$. Deduce that the right hand side of (1) can be found by solving the linear programming problem

$$\underset{s_i,t_j}{\text{maximize}} \Big\{ \sum_i s_i + \sum_j t_j : \alpha_i\beta_j - s_i - t_j \geq 0,\ \forall i,j \Big\}. \tag{3}$$

The Kronecker product matrix $A \otimes C$ has all its eigenvalues of the form $\alpha_i\beta_j$. In can be shown that (3) can be written as the semidefinite programming problem:

$$\underset{S,T}{\text{maximize}} \Big\{ \mathrm{tr}(S) + \mathrm{tr}(T) : A\otimes C - S\otimes I - I\otimes T \succeq 0 \} \tag{4}$$

where maximization is over symmetric matrices $S, T$. How might this be used in an algorithm to solve (1)?