Mastermath and LNMB Course: Discrete Optimization

Solutions for the Exam 9 January 2012

Utrecht University, Educatorium, 15:15–18:15

The examination lasts 3 hours. Grading will be done before January 23, 2012. Students interested in checking their results can make an appointment by e-mail (g.schaefer@cwi.nl).

The examination consists of six problems. The maximum number of points to be gained on the different parts are displayed in the following table:

1(a)–(j)	2	3(a)	3(b)	3(c)	4	5	6(a)	6(b)	6(c)	$\sum$
20 (2 each)	10	5	5	5	15	15	5	10	10	**100**

The grade for the exam is obtained by dividing the total number of points by **10**. This implies that **55** points are needed to pass.

During the examination only the Lecture Notes of the course without any additional leaflets are allowed to be on your desk and all electronic equipment must be switched off.

Please be short, clear and precise in your answers. If you use results from the Lecture Notes, please provide the respective references. Please hand in your answers **together** with the exam sheet.

Wishing you a Happy New Year 2012 and Good Luck!

Problem 1 (20 points (2 points each))**.** State for each of the claims below whether it is **true** of **false**. Note: You need not to justify or proof your answers here.

(a) $\sqrt{n} = \Omega(n)$.

(b) Let $G = (V,E)$ be a bipartite graph and define $\mathcal{I} = \{M \subseteq E \mid M \text{ is a matching of } G\}$. Then $(E,\mathcal{I})$ is a matroid.

(c) Given a spanning tree T of a graph $G = (V,E)$, let $\deg_T(u)$ refer to the number of edges in T that are incident to $u \in V$. Then $\sum_{u \in V} \deg_T(u) = 2n-1$.

(d) A graph is bipartite if and only if it does not contain an odd length cycle.

(e) Given a directed graph $G = (V,E)$ with edge costs $c : E \to \mathbb{R}$, one can determine in polynomial time whether G contains a cycle of negative cost.

(f) If $\Pi_1 \in P$ and $\Pi_2 \preceq \Pi_1$, then $\Pi_2 \in P$.

(g) If Π_1 is *NP*-complete and $\Pi_1 \preceq \Pi_2$, then Π_2 is *NP*-complete.

(h) If there exists an *NP*-complete problem Π such that $\Pi \in P$, then $P = NP$.

(i) If there is a pseudo-polynomial time algorithm for an optimization problem Π, then $\Pi \in P$.

(j) There exists a $\frac{3}{2}$-approximation algorithm for *TSP*.

Solution:

(a) false

(b) false

(c) false

(d) true

(e) true

(f) true

(g) false

(h) true

(i) false

(j) false (unless P = NP)

Problem 2 (10 points)**.** Let $M_1 = (S,\mathcal{I}_1)$ and $M_2 = (S,\mathcal{I}_2)$ be two matroids over the same ground set S. The *intersection* $M_1 \cap M_2$ of M_1 and M_2 is defined as the independent set system $(S,\mathcal{I}_1 \cap \mathcal{I}_2)$. Let $G = (L \cup R, E)$ be a bipartite graph and define

$$\mathcal{I} = \{M \subseteq E \mid M \text{ is a matching of } G\}.$$

Show that the independent set system $(E,\mathcal{I})$ is the intersection of two matroids. (Hint: An *L-sided matching* of G is a subset $M \subseteq E$ such that each node $u \in L$ has at most one edge in M incident to it.)

Solution: *We first show that the pair $(E, \mathcal{I}_L)$ with*

$$\mathcal{I}_L = \{M \subseteq E \mid M \text{ is an L-sided matching of } G\}$$

constitutes a matroid. First, note that $\emptyset \in \mathcal{I}_L$. Second, let $M \in \mathcal{I}_L$ and $M' \subseteq M$. Clearly, because M is an L-sided matching, every node $u \in L$ has at most one edge in M' incident to it. Thus $M' \in \mathcal{I}_L$. Finally, let $M, M' \in \mathcal{I}_L$ and assume $|M'| < |M|$. Then there is a node $u \in L$ that is matched in M and unmatched in M'. By adding the matching edge $e \in M$ incident to u to M' we obtain a new L-sided matching $M + e \in \mathcal{I}_L$. Thus, $(E, \mathcal{I}_L)$ is a matroid by Definition 3.2 of the Lecture Notes.

We proceed exactly the same way to show that $(E, \mathcal{I}_R)$ with

$$\mathcal{I}_L = \{M \subseteq E \mid M \text{ is an R-sided matching of } G\}$$

*is a matroid. (An R-*sided matching *of G is a subset $M \subseteq E$ such that each node $u \in R$ has at most one edge in M incident to it.)*

Finally, we prove that $M \subseteq E$ is a matching of G if and only if $M \in \mathcal{I}_L \cap \mathcal{I}_R$. Clearly, if M is a matching then M is both an L-sided and an R-sided matching by definition. To show the converse, suppose for the sake of a contradiction that $M \in \mathcal{I}_L \cap \mathcal{I}_R$ and M is not a matching. Because M is not a matching, there must two edges e_1 and e_2 that share a common endpoint, say u. Suppose $u \in L$ (the case $u \in R$ follows analogously). But then two edges are incident to u and M is thus not a left-sided matching, which is a contradiction to the assumption that $M \in \mathcal{I}_L$.

Problem 3 ($5+5+5$ points)**.** In the *minimum-weight edge cover problem* we are given an undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}^+$. An *edge cover* $E' \subseteq E$ of G is a subset of the edges such that each node $u \in V$ is *covered* by at least one edge in E', i.e., for every node $u \in V$ there is an edge $(u, v) \in E'$. The goal is to compute an edge cover E' of minimum total weight $w(E') = \sum_{e \in E'} w(e)$.

(a) Formulate the *minimum-weight edge cover problem* as an integer linear program and derive the respective LP relaxation.

(b) Show that the set of feasible solutions of this LP is an integral polytope if G is bipartite.

(c) Give an example that shows that an optimal solution to this LP might be non-integral if G is not bipartite.

Solution:

(a) The integer programming formulation of the minimum-weight edge cover problem is as follows: We have an indicator variable $x_e \in \{0, 1\}$ for every edge $e \in E$ with $x_e = 1$

iff e belongs to the edge cover E'.

$$\begin{aligned} \textit{minimize} \quad & \sum_{e \in E} w(e) x_e \\ \textit{subject to} \quad & \sum_{e=(u,v) \in E} x_e \geq 1 \qquad \forall u \in V \\ & x_e \in \{0,1\} \quad \forall e \in E \end{aligned} \tag{1}$$

The LP relaxation is

$$\begin{aligned} \textit{minimize} \quad & \sum_{e \in E} w(e) x_e \\ \textit{subject to} \quad & \sum_{e=(u,v) \in E} x_e \geq 1 \quad \forall u \in V \\ & x_e \geq 0 \quad \forall e \in E \end{aligned} \tag{2}$$

Note that the constraints "$x_e \leq 1 \; \forall e \in E$" are redundant because of the minimization objective.

(b) Consider LP (2)*. We can rewrite this LP as follows:*

$$\min w^T x \quad \textit{s.t.} \quad Ax \geq 1, \; x \geq 0$$

where A is the incidence matrix of an undirected bipartite graph. By Corollary 8.1 of the Lecture Notes, A is totally unimodular. As a consequence, $-A$ is also totally unimodular (multiplying by -1 does not change the absolute value of the determinant). Also, -1 is integral and thus the polytope $P = \{x \mid -Ax \leq -1, \; x \geq 0\}$ is integral by Theorem 8.8 of the Lecture Notes.

(c) Consider a cycle of length 3 with every edge having unit weight. An optimal solution to LP (2) *is $x_e = \frac{1}{2}$ for every edge $e \in E$ with total weight $\frac{3}{2}$. This solution is not integral.*

Problem 4 (15 points)**.** Consider the following *minimum shortest-path-cut problem*: We are given a directed graph $G = (V,E)$ with edge costs $c : E \to \mathbb{R}^+$, a source node $s \in V$ and a target node $t \in V$. The goal is to find a subset $C \subseteq E$ of minimum cardinality such that every shortest s,t-path P (with respect to c) is *cut* by C, i.e., $P \cap C \neq \emptyset$.

Derive an algorithm that solves this problem, prove its correctness and analyze its running time.

Solution: *The algorithm is as follows:*

First note that the algorithm terminates. The correctness of the algorithm follows from the following lemma:

Lemma 1. *P is a shortest s,t-path in G if and only if P is an s,t-path in G'.*

Proof. First note that the distance function δ is well defined because edge costs are non-negative. Suppose P is a shortest s,t-path in G. Then all edges of P must be tight

Input: A directed graph $G = (V,E)$ with edge costs $c : E \to \mathbb{R}^+$, a source node $s \in V$ and a target node $t \in V$.
Output: Subset $C \subseteq E$ of minimum size that cuts every shortest s,t-path.

1 Solve the SSSP problem with s as source node to determine distances $\delta(u)$ for every $u \in V$.
2 Construct the subgraph of G that contains all tight edges with respect to δ. Remove from this graph all edges that do not lie on an s,t-path. Let the resulting graph be $G' = (V,E')$.
3 Compute a minimum capacity s,t-cut $(X,\bar{X})$ (with respect to unit capacities) of G' and let C be the set of directed edges from X to $\bar{X}$.
4 Output C.

(Lemma 4.4. of the Lecture Notes) and thus P is part of G. Next consider an arbitrary s,t-path $P = \langle s = v_1, v_2, \ldots, v_k = t \rangle$ in G'. The cost of this path is

$$c(P) = \sum_{i=1}^{k-1} c(v_i, v_{i+1}) = \sum_{i=1}^{k-1} \delta(v_{i+1}) - \delta(v_i) = \delta(t) - \delta(s) = \delta(t).$$

Thus P is a shortest s,t-path. □

In light of Lemma 1, the goal now is to determine a subset of edges $C \subseteq E'$ of minimum cardinality such that after removing C form G', s and t are disconnected in G'. Let $(X,\bar{X})$ be a minimum capacity s,t-cut of G', where each edge $e \in E'$ has unit capacity. Define $C = \{(u,v) \in E' \mid u \in X,\ v \in \bar{X}\}$. By Theorem 5.3 of the Lecture Notes, C cuts every s,t-path in G' and has minimum size.

We next analyze the running time of the algorithm: We can use Dijkstra's algorithm in Step 1 because edge costs are non-negative. This takes $O(m + n\log n)$ time. Step 2 requires that we check for every edge $e \in E$ whether e is tight. This takes time $O(m)$. We can determine (i) all nodes that are reachable from s and (ii) all nodes from which t is reachable in time $O(n+m)$ by running one depth-first search from s and one depth-first search from t (on the graph with all edge directions reversed). Step 2 thus takes $O(n+m)$ time in total. Computing a minimum capacity s,t-cut in Step 3 can be done by a max-flow computation (see Theorem 5.3 of the Lecture Notes) which takes time $O(nm^2)$ if we use the Edmonds-Karp algorithm. Identifying the edges in the cut requires $O(n+m)$ time. The overall running time of the algorithm is dominated by the max-flow computation and is thus $O(nm^2)$.

Problem 5 (15 points)**.** The decision variant of the *knapsack problem* is as follows: We are given a set $N = \{1,\ldots,n\}$ of n items with each item $i \in N$ having a profit $p_i \in \mathbb{Z}^+$ and a weight $w_i \in \mathbb{Z}^+$, a knapsack capacity $B \in \mathbb{Z}^+$ and a parameter $K \in \mathbb{Z}^+$. The goal is to determine whether there exists a subset $X \subseteq N$ such that $w(X) = \sum_{i \in X} w_i \leq B$ and $p(X) = \sum_{i \in X} p_i \geq K$.

Prove that the *knapsack problem* is *NP*-complete. (Hint: Use that the following *subset*

sum problem is *NP*-complete: Given n non-negative integers $s_1, \ldots, s_n$ and a parameter L, determine whether there is a subset of these numbers whose total sum is L.)

Solution: *We first argue that knapsack $\in$ NP. A certificate of a yes-instance is a subset X of N with $w(X) \leq B$ and $p(X) \geq K$. Both conditions can be checked in linear time (in n).*

In order to prove NP-completeness, we show that subset sum $\preceq$ knapsack. The claim then follows because the subset sum problem is NP-complete. Let $I = (s_1, \ldots, s_n, L)$ be an instance of the subset sum problem. Define an instance I' of the knapsack problem as follows: $N = \{1, \ldots, n\}$ and for every $i \in N$: $p_i = w_i = s_i$. Moreover, let $B = K = L$. This transformation takes linear time (in n).

We claim that I is a yes-instance of the subset sum problem iff I' is a yes-instance of the knapsack problem. Let $S \subseteq N$ such that $\sum_{i \in S} s_i = L$. Then the weight and profit of the corresponding knapsack $X = S$ is $w(X) = B$ and $p(X) = K$, respectively. Thus X is a yes-instance of the knapsack problem. Let $X \subseteq N$ be a yes-instance of the knapsack problem. Then $w(X) \leq B$ and $p(X) \geq K$. Because $p_i = w_i = s_i$ for all items i and $B = K = L$, we have $w(X) = p(X) = L$. Thus, with $S = X$, $\sum_{i \in S} s_i = L$ and thus I is a yes-instance of the subset sum problem.

Problem 6 ($5 + 10 + 10$ points)**.** A natural greedy algorithm to compute a solution for the *knapsack problem* is as follows:

Input: A set $N = \{1, \ldots, n\}$ of items with a profit $p_i \in \mathbb{Z}^+$ and a weight $w_i \in \mathbb{Z}^+$ for every item $i \in N$ and a knapsack capacity $B \in \mathbb{Z}^+$.
Output: Subset $X \subseteq N$ of items.

1 Sort the items by non-increasing profit per weight ratio and re-index the items such that
$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}.$$
2 Let k be the smallest index such that $\sum_{i=1}^{k+1} w_i > B$.
3 Output $X = \{1, \ldots, k\}$.

(a) Give an example that shows that the approximation ratio of the above algorithm can be arbitrarily large.

(b) Let OPT be the profit of an optimal solution. Prove that the following relation holds:
$$\text{OPT} \leq \sum_{i=1}^{k} p_i + \frac{B - \sum_{i=1}^{k} w_i}{w_{k+1}} p_{k+1}.$$
(Hint: Formulate the *knapsack problem* as an integer linear program, derive the LP relaxation and characterize its optimal solutions.)

(c) Obtain a 2-approximation algorithm for the *knapsack problem* by adapting the above algorithm and show that the approximation ratio is tight.

Solution:

(a) Suppose we have two items with $p_1 = 2$ and $w_1 = 1$ and $p_2 = B = w_2$. The above greedy algorithm picks item 1 with a total profit of 2. An optimal solution picks item 2 with total profit B. The approximation ratio $B/2$ becomes arbitrarily large as $B \to \infty$.

(b) The following is a natural integer linear programming formulation of the knapsack problem:

$$\begin{aligned} \textit{maximize} \quad & \sum_{i=1}^{n} p_i x_i \\ \textit{subject to} \quad & \sum_{i=1}^{n} w_i x_i \leq B \\ & x_i \in \{0,1\} \quad \forall i \in \{1,\ldots,n\} \end{aligned} \tag{3}$$

The LP relaxation of (3) *is:*

$$\begin{aligned} \textit{maximize} \quad & \sum_{i=1}^{n} p_i x_i \\ \textit{subject to} \quad & \sum_{i=1}^{n} w_i x_i \leq B \\ & 0 \leq x_i \leq 1 \quad \forall i \in \{1,\ldots,n\} \end{aligned} \tag{4}$$

Note that the constraints "$x_i \leq 1 \; \forall i \in \{1,\ldots,n\}$" are not redundant here.
Let OPT *and* OPT_{LP} *refer to the total profit of an optimal solution for* (3) *and* (4)*, respectively. Because* (4) *is the LP relaxation of* (3)*, we have* $\text{OPT} \leq \text{OPT}_{LP}$.
We claim the following:

Lemma 2. *The following solution x^* is an optimal solution to LP* (4)*:*

$$x_i^* = \begin{cases} 1 & \text{if } i \in \{1,\ldots,k\} \\ \frac{B - \sum_{i=1}^{k} w_i}{w_{k+1}} & \text{if } i = k+1 \\ 0 & \text{otherwise} \end{cases}$$

Proof. Clearly, $0 \leq x_i^* \leq 1$ for every $i \in N$ and the total weight of the solution x^* is $\sum_{i \in N} w_i x_i^* = B$. Thus, x^* is feasible. Moreover, x^* is an optimal solution because the items are chosen according to non-increasing profit per weight ratio (this follows from an exchange argument). □

We conclude

$$\text{OPT} \leq \text{OPT}_{LP} = \sum_{i=1}^{n} p_i + \frac{B - \sum_{i=1}^{k} w_i}{w_{k+1}} p_{k+1}.$$

(c) We adapt the given algorithm by returning the more profitable set of items among $\{1,\ldots,k\}$ and $\{k+1\}$.
Note that the algorithm outputs a feasible solution (we assume that $w_i \leq B$ for every $i \in N$). Moreover, the algorithm has polynomial running time: Sorting the items by their profit per weight ratio takes $O(n \log n)$. The index k in Step 2 can be determined in time $O(n)$. Altogether the algorithm needs $O(n \log n)$ time.
It remains to be shown that the algorithm achieves an approximation ratio of 2:

The total profit of the solution X output by the algorithm satisfies:

$$2p(X) \geq \sum_{i=1}^{k} p_i + p_{k+1} \geq \text{OPT}.$$

Here the second inequality follows from part (b) because

$$\text{OPT} \leq \sum_{i=1}^{n} p_i + \frac{B - \sum_{i=1}^{k} w_i}{w_{k+1}} p_{k+1} \leq \sum_{i=1}^{n} p_i + p_{k+1}.$$

The claim follows.
The following example shows that the approximation ratio of 2 is tight: Consider three items with $p_1 = 2$ and $w_1 = 1$ and $p_2 = p_3 = \frac{1}{2}B = w_2 = w_3$. The algorithm selects items 1 and 2 of total profit $\frac{1}{2}B + 2$. An optimal solution chooses items 2 and 3 of total profit B. Thus the approximation ratio for this instance is $B/(\frac{1}{2}B + 2) \to 2$ as $B \to \infty$.