Mastermath and LNMB Course: Discrete Optimization

Solutions for the Exam 6 January 2014

Utrecht University, Educatorium, 13:30–16:30

The examination lasts 3 hours. Grading will be done before January 20, 2014. 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 as indicated below.

1(a)–(j)	2	3	4	5	6(a)	6(b)	$\sum$
20 (2 each)	10	15	10	10	15	10	**90**

The grade for the exam is obtained by dividing the total number of points by **9**. This implies that **49.5** points are needed to pass.

During the exam, only the Lecture Notes and the assignments (incl. solutions) are allowed to be on your desk and all electronic equipment must be switched off. **No additional material is permitted**. The Lecture Notes may contain annotations that you made during the lectures. You may use the assignments with the solutions that were provided (the solutions that you handed in are not allowed).

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 2014 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 justify or prove your answers here.

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

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

(c) Let $M = (S,\mathcal{I})$ be a matroid and k a natural number. Define $\mathcal{I}' = \{I \in \mathcal{I} \mid |I| \leq k\}$. Then $M' = (S,\mathcal{I}')$ is a matroid.

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

(e) Let $G = (V,E)$ be a directed graph with capacities $c : E \to \mathbb{Q}^+$ and source and destination nodes $s,t \in V$. If f is a preflow and $\sum_{u \in V \setminus \{s,t\}} e_f(u) = 0$ then f is a flow.

(f) Let M^* be a maximum cardinality matching of an undirected graph $G = (V,E)$. If M is a matching of G and $k = |M^*| - |M| \geq 1$ then the shortest M-augmenting path contains at most $|M|/k$ edges from M.

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

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

(i) The *metric TSP problem* is strongly *NP*-complete.

(j) The *minimum spanning tree problem* is a special case of the *Steiner tree problem*.

Solution:

(a) false

(b) false

(c) true

(d) true

(e) true

(f) true

(g) true

(h) false

(i) true

(j) true

Grading scheme: *2 points for each correct answer.*

Additional remarks:

- *It was not required to justify the answers, so any proofs were ignored.*
- *Some students argued that (j) is wrong because the minimum spanning tree problem admits negative costs, whereas the Steiner tree problem only admits nonnegative*

costs. However, the MST problem with arbitrary edge costs can be reduced to the MST problem with non-negative costs by adding a suitably large constant to each edge cost (see also Assignment 1, Problem 2).

Problem 2 (10 points)**.** Consider the following problem:

Allocation Problem:

Given: A bipartite graph $G = (X \cup Y, E)$ with edge weights $w : E \to \mathbb{Q}^+$.
Goal: Find a subset $M \subseteq E$ such that each node in X is incident to at most one edge in M and the total weight $\sum_{e \in M} w(e)$ is maximized.

Show that the *allocation problem* can be solved by the greedy algorithm for matroids.

Solution: *Call a set $M \subseteq E$ that satisfies that each node $u \in X$ has at most one edge from M incident to it an X-*sided matching*. We show that the pair $(E, \mathcal{I})$ with*

$$\mathcal{I} = \{M \subseteq E \mid M \text{ is an } X\text{-sided matching}\}$$

constitutes a matroid. The claim then follows by Theorem 3.1 of the Lecture Notes.

1. *First, note that $\emptyset \in \mathcal{I}$.*
2. *Second, let $M \in \mathcal{I}$ and $M' \subseteq M$. Clearly, because M is an X-sided matching, every node $u \in X$ has at most one edge in M' incident to it. Thus $M' \in \mathcal{I}$.*
3. *Finally, let $M, M' \in \mathcal{I}$ and assume $|M'| < |M|$. Let $X(M)$ refer to the set of nodes in X that are incident to some edge in M, i.e.,*

   $$X(M) = \{u \in X \mid \exists \{u, v\} \in M\}.$$

   $X(M')$ is defined analogously. Note that because G is bipartite and both M' and M are X-sided matchings we have $|M'| = |X(M')|$ and $|M| = |X(M)|$. By assumption $|M'| < |M|$ and thus there is a node $u \in X$ 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 X-sided matching $M + e \in \mathcal{I}$. Thus, $(E, \mathcal{I})$ is a matroid by Definition 3.2 of the Lecture Notes.

Grading scheme:

- *2 points for defining the independent set system*
- *1 point for proving Property 1*
- *2 points for proving Property 2*
- *5 points for proving Property 3*

Additional remarks: *Most students got full points here.*

Problem 3 (15 points)**.** Consider the following problem:

Hit-All-Shortest-Paths Problem:

Given: A directed graph $G = (V,E)$ with edge costs $c : E \to \mathbb{Q}^+$, a source node $s \in V$ and a target node $t \in V$.

Goal: Find a subset $C \subseteq E$ of minimum cardinality such that every shortest s,t-path P (with respect to c) is *hit* 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 described below.*

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.

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 (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 5.6 of the Lecture Notes) which takes time $O(n^2m)$ if we use the preflow-push algorithm. Identifying the edges in the cut requires $O(m)$ time. The overall running time of the algorithm is dominated by the max-flow computation and is thus $O(n^2m)$.

Grading scheme:

- *6 points for the algorithm, consisting of:*
  - *2 for each of Steps 1, 2, 3 (see algorithm)*
- *6 points for correctness, consisting of:*
  - *1 point for termination*
  - *1 point for arguing explicitly that the shortest paths in G are exactly the paths in G'*
  - *2 points for arguing that C hits all shortest paths*
  - *2 points for showing that C is minimal*
- *3 points for running time analyses, consisting of:*
  - *1 point for running time of Dijkstra*
  - *1 point for running time of finding the subgraph*
  - *1 point for running time of finding the cut*

Problem 4 (10 points)**.** Consider the following problem:

Set Cover Problem:

Given: A set $U = \{1,\dots,n\}$ of n elements, a collection of m subsets $S_1,\dots,S_m \subseteq U$ and a natural number K.

Goal: Determine whether there is a selection of at most K subsets such that their union is U, i.e., whether there exists some $I \subseteq \{1,\dots,m\}$ with $|I| \leq K$ such that $\bigcup_{i\in I} S_i = U$.

Prove that the *set cover problem* is *NP*-complete.

Solution: *We first argue that set cover $\in$ NP. A certificate of a yes-instance is a subset $I \subseteq \{1,\dots,m\}$ with $|I| \leq K$ such that $\bigcup_{i\in I} S_i = U$. Both conditions can be checked in time $O(nm)$.*

In order to prove NP-completeness, we show that vertex cover $\preceq$ set cover. The claim then follows because the vertex cover problem is NP-complete. Let $\mathcal{I} = (G,K)$ be an instance of the vertex cover problem*, where $G = (V,E)$ is an undirected graph and K is a parameter (for the size of the vertex cover). For notional convenience let the node set be $V = \{u_1,\ldots,u_m\}$ and the edge set $E = \{e_1,\ldots,e_n\}$ (note that m denotes the number of nodes and n the number of edges of G here). Define an instance $\mathcal{I}'$ of the set cover problem as follows: Let $U = \{1,\ldots,n\}$ and for every node $u_i \in V$, $i \in \{1,\ldots,m\}$, define*

$$S_i = \{j \in U \mid \text{edge } e_j \text{ is incident to node } u_i \text{ in } G\}.$$

The interpretation here is that the set of elements U to be covered corresponds to the edge set E and that each set S_i, $i \in \{1,\ldots,m\}$, represents the set of edges covered by picking node $u_i \in V$. The parameter K of the set cover instance is the same as for the vertex cover instance. This transformation takes linear time $O(n+m)$.

We prove that $\mathcal{I}$ is a yes-instance of the vertex cover problem iff $\mathcal{I}'$ is a yes-instance of the set cover problem. Let $C \subseteq V$ be a vertex cover of G satisfying that $|C| \leq K$. Then, because C is a vertex cover and by the construction of the instance $\mathcal{I}'$, the set $I \subseteq \{1,\ldots,m\}$ with $i \in I$ iff $u_i \in C$ satisfies $\bigcup_{i\in I} S_i = U$. Further, $|I| = |C| \leq K$ and thus I is a yes-instance of the set cover problem*. Let $I \subseteq \{1,\ldots,m\}$ with $|I| \leq K$ be a yes-instance of the set cover problem. Then by the construction of $\mathcal{I}'$, all edges of G are covered by the vertices in $C = \{u_i \in V \mid i \in I\}$. Thus, C is a vertex cover of cardinality $|C| = |I| = K$, i.e., C is a yes-instance of the vertex cover problem.*

Grading scheme:

- *2 points for showing that set cover $\in$ NP*
- *3 points for the correct transformation of the instance from vertex cover*
- *1 point for arguing that this reduction can be done in polynomial time*
- *4 points for proving that yes-instances correspond to each other (2 points for each direction)*

Additional remarks:

- *Many students claimed that verification of a yes-instance of set cover can be done in $O(m)$ or $O(m+n)$. However, the obvious way to determine the union of m sets (each containing at most n elements) requires $O(nm)$ time. Nevertheless, no points were deducted for this minor mistake.*
- *Some students realized that a reduction from vertex cover is the way to go but defined a subset as the set of vertices that are adjacent to an edge, which is unfortunately wrong.*
- *A few students tried to reduce from satisfiability. The clauses are the elements in the ground sets and the literals $x_1,\ldots,x_n,\neg x_1,\ldots,\neg x_n$ represent the subsets. Every subset contains the clauses in which the literal is contained. This idea is nice, but one direction in the equivalence proof of yes-instance does not work: Once one has a yes-instance this could require that one would have to put both x_3 and $\neg x_3$ to* TRUE*, which is impossible. Still a reasonable number of points were awarded for*

the good attempt.

Problem 5 (10 points)**.** Consider the decision variant of the *knapsack problem*:

Knapsack Problem:

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 natural number K.

Goal: 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.

Grading scheme:

- *2 points for showing that knapsack $\in$ NP*
- *3 points for the correct transformation of the instance from subset sum*
- *1 point for arguing that this reduction can be done in polynomial time*
- *4 points for proving that yes-instances correspond to each other (2 points for each direction)*

Additional remarks:

- *Most students got full points here.*
- *Once you have the correct transformation from subset sum ($p_i = w_i = s_i$), showing*

that the yes-instances of the two problems are equivalent is not very difficult. However, this does not mean you can simply write "obvious", because there are some minor insights that needed to be used (e.g., argue that $\sum_{i\in S} s_i \leq L$ and $\sum_{i\in S} s_i \geq L$ implies that $\sum_{i\in S} s_i = L$).

- *Giving a correct reduction from partition also gave 10 points.*

Problem 6 ($15+10$ points)**.** Consider the *multiway cut problem*:

Multiway Cut Problem:

Given: A connected, undirected graph $G=(V,E)$ with edge weights $w : E \to \mathbb{Q}^+$ and a set $T = \{t_1,\ldots,t_k\} \subseteq V$ of *terminals*.

Goal: Find a subset $E' \subseteq E$ of minimum total weight $w(E') = \sum_{e\in E'} w(e)$ whose removal disconnects all terminals in T from each other.

(a) Derive a 2-approximation algorithm for the *multiway cut problem*. (Hint: Combine k minimum weight cuts $C_1,\ldots,C_k$, where C_i separates t_i from the rest of the terminals.)

(b) Refine your algorithm to obtain a $(2-2/k)$-approximation algorithm and provide an example that shows that this approximation ratio is tight.

Solution:

(a) We call a cut $C_i \subseteq E$ whose removal separates terminal t_i from the remaining terminals $T \setminus \{t_i\}$ a t_i-isolating cut.

Consider the following algorithm:

> 1 *Compute a minimum weight t_i-isolating cut C_i for every $i = 1,\ldots,k$.*
> 2 *Output $E' = \cup_{i=1}^k C_i$.*

The minimum weight t_i-isolating cut C_i can be computed by solving a minimum cut problem: Let $s = t_i$, add a artificial target node t and connect all terminals in $T \setminus \{t_i\}$ to t. Then compute a minimum weight s,t-cut $(X_i,\bar{X}_i)$ in the resulting graph. Every such cut can be computed in time $O(n^2m)$ (using the preflow-push algorithm), where n and m refer to the number of nodes and edges, respectively. Extracting the respective edge set C_i crossing $(X_i,\bar{X}_i)$ can be done in time $O(m)$. The overall running time is thus at most $O(kn^2m) = O(n^3m)$, which is polynomial.

Observe that the solution output by the algorithm is feasible because C_i separates t_i from $T \setminus \{t_i\}$ and we combine all these cuts.

We next show that $\text{OPT} \geq \frac{1}{2}\sum_{i=1}^k w(C_i)$. *Consider an optimal multiway cut C^*. By removing C^* from G, we obtain k connected components each containing exactly one terminal. (Note that because C^* is a minimum weight cut, no more than k components*

will be created.) Let C_i^* be the subset of edges in C^* that separate the component containing terminal t_i from the rest. Then $C^* = \cup_{i=1}^k C_i^*$. Note that every edge $e \in C^*$ belongs to exactly two isolating cuts C_i^*, C_j^* for some $i \neq j$. Thus $\sum_{i=1}^k w(C_i^*) = 2w(C^*) = 2\text{OPT}$.

Observe that $w(C_i) \leq w(C_i^*)$ because C_i is a minimum weight t_i-isolating cut. Thus

$$w(E') \leq \sum_{i=1}^{k} w(C_i) \leq \sum_{i=1}^{k} w(C_i^*) = 2\text{OPT}.$$

The algorithm therefore produces a feasible multiway cut of total weight $w(E') \leq 2\text{OPT}$.

Grading scheme:

- *3 points for stating the algorithm;*
- *2 points for arguing that the algorithm needs polynomial time;*
- *2 points for arguing that the algorithm outputs a feasible solution;*
- *4 points for proving a lower bound on* OPT*;*
- *4 points for relating the solution output by the algorithm to* OPT.

(b) Consider the following refinement of the algorithm:

1 *Compute a minimum weight t_i-isolating cut C_i for every $i = 1, \ldots, k$.*
2 *Let C_h be a cut of maximum weight among $C_1, \ldots, C_k$*
3 *Output $E' = \cup_{i \neq h} C_i$.*

As before, the algorithm has polynomial running time.

We show that the final subset E' is a feasible solution to the multiway cut problem. Suppose for the sake of a contradiction that after removing E' two terminals $t_i, t_j \in T$ remain connected. Because we only discard the maximum weight cut, at least one of the two isolating cuts C_i and C_j must be part of E', say $C_i \subseteq E'$. But then E' separates t_i from $t_j \in T \setminus \{t_i\}$, which is a contradiction.

Finally, observe that because we exclude the heaviest of the k cuts we obtain

$$w(E') \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^{k} w(C_i) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^{k} w(C_i^*) = \left(2 - \frac{2}{k}\right) \text{OPT}.$$

The following is a tight example. Take a cycle consisting of k inner nodes with each edge having a weight of 1. Connect each inner node by an edge to an outer terminal node and assign a weight of $2 - \varepsilon$ to this edge.

The above algorithm outputs $k-1$ edges of weight $2-\varepsilon$ and thus $w(E') = (k-1)(2-\varepsilon)$. On the other hand, removing all edges of the cycle has weight k and thus

$\text{OPT} \leq k$. *The approximation ratio of the algorithm is thus at least* $(2-\varepsilon)(k-1)/k = (2-\varepsilon)(1-\frac{1}{k})$ *which approaches* $2-\frac{2}{k}$ *as* $\varepsilon \to 0$.

Grading scheme:

- *2 points for stating the refined algorithm;*
- *3 points for arguing about polynomial running time, feasibility and improved approximation ratio;*
- *5 points for giving a tight example.*

Additional remarks: *Problem 6b turned out to be too hard and we therefore decided to turn it into a problem where one could gain 10 "bonus points".*