Approximation Algorithms

Winter term 2021/22

Prof. Dr. Martin Hoefer
Tim Koglin, Lisa Wilhelmi

GOETHE UNIVERSITÄT FRANKFURT AM MAIN
Institute of Computer Science
Algorithms and Complexity

Assignment 2

Issued: 02.11.2021
Due: 09.11.2021, **10:15h**

Exercise 2.1 *Greedy Variants for* INTERVALSCHEDULING (2 + 2 + 2 points)

In the lecture, the *GreedyIntervals* algorithm was discussed. It solves the INTERVALSCHEDULING problem optimally by operating as follows: The algorithm sorts all tasks $t_i \in T$, for $i \in [n]$, in non-decreasing order with respect to their deadlines $d_i > 0$. This yields a sorted list $A = [a_1, a_2, ..., a_n]$, where $a_1 := \arg\min_{t_i \in T} d_i$. In the next step, the algorithm iterates over all tasks in A consecutively starting with a_1. A task is added to the solution set if it does not collide with any task that has already been added to it.

Consider the following modifications M_1 and M_2 of *GreedyIntervals*:

- M_1: The tasks are sorted with respect to their release dates $r_i > 0$ in non-increasing order. This yields a sorted list $B = [b_1, b_2, ..., b_n]$, where $b_1 := \arg\max_{t_i \in T} r_i$. In the next step, the algorithm iterates over all tasks in B consecutively starting with b_1.
- M_2: Consider a combination of the original *GreedyIntervals* algorithm and M_1. It generates both lists A and B (as defined above). Now, the iteration is over the elements of the lists in an alternating fashion starting with a_1, i.e., $a_1, b_1, a_2, b_2, a_3, b_3, ...$.

a) Determine the solutions obtained by each of the algorithms on the instance for INTERVALSCHEDULING with $T = \{t_1, t_2, t_3, t_4, t_5\}$ given in the following table. For each solution, also state whether or not it is optimal for this instance.

	t_1	t_2	t_3	t_4	t_5
r_i	1	2	4	6	7
d_i	4	6	5	9	10

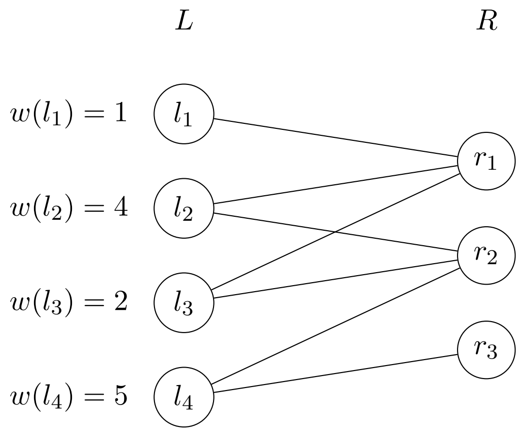
b) Investigate whether or not M_1 computes an optimal solution for INTERVALSCHEDULING in general.

c) Investigate whether or not M_2 computes an optimal solution for INTERVALSCHEDULING in general.

Exercise 2.2 *Greedy-optimal* MaxMatching (2 + 3 points)

Let $G = ((L \cup R), E)$ be an undirected and bipartite graph, i.e., $L \cap R = \emptyset$ and every edge $e \in E$ connects a vertex in L to a vertex in R. Further, each vertex $l_i \in L$ has a weight $w(l_i) \geq 0$. Consider the set system $(\mathcal{P}, L)$. Here, $X \in \mathcal{P}$ if there exists a matching M of G such that at least all vertices in X are incident to an edge in M.

a) Determine all non-extendable sets of $\mathcal{P}$ in the following instance. No explanation is required.



b) Show that $(\mathcal{P}, L)$ is a matroid.

Exercise 2.3 *Intersecting Matroids* (4 points)

Let $G = ((L \cup R), E)$ be an undirected, bipartite graph (see definition in Exercise 2.2). In the downward-closed set system $(\mathcal{M}, E)$, the set $\mathcal{M}$ contains all possible matchings of the edges in E.

Show that $(\mathcal{M}, E)$ can be represented as the intersection of two matroids.

Exercise 2.4 MaxMinMatroid *Problem* (2 + 3 points)

In the MaxMinMatroid problem, a matroid $(\mathcal{I}, \mathcal{R})$ is given, where $\mathcal{B} \subseteq \mathcal{I}$ is the set of all its bases. The goal is to compute a basis $X^* \in \mathcal{B}$ with highest minimum value, i.e.,

$$X^* \in \arg\max_{X \in \mathcal{B}} \min_{e \in X} w(e).$$

a) Determine a basis X^* of the matroid given in Exercise 2.2 a). No explanation is required.

b) Show that the *Greedy* algorithm (Algorithm 5 from the lecture notes) solves the MaxMin-Matroid problem optimally for $(\mathcal{I}, \mathcal{R})$.
Hint: Use the extendability property and greedy optimality for sum of weights.

The assignments and further information on the course are provided on our website:
https://algo.cs.uni-frankfurt.de/lehre/apx/winter2122/apx2122.shtml

Contacts: {koglin,wilhelmi}@em.uni-frankfurt.de.