Technische Universität München, Zentrum Mathematik
Lehrstuhl für Angewandte Geometrie und Diskrete Mathematik

Combinatorial Optimization (MA4502)
SS 2016

Prof. Dr. Andreas S. Schulz

Exam (English Summary)

Problem 1

Let V be a finite set, $N \geq 1$, and let $h : V \to \{0, 1, \ldots, N\}$.

(a) For $U \subseteq V$ define

$$f(U) := \begin{cases} \max\{h(u) \,:\, u \in U\} & : \quad U \neq \emptyset, \\ 0 & : \quad U = \emptyset. \end{cases}$$

Is f submodular? (Give a proof or counterexample.)

(b) For $U \subseteq V$ define

$$g(U) := \begin{cases} \max\{h(u) \,:\, u \in U\} & : \quad U \neq \emptyset, \\ 1 & : \quad U = \emptyset. \end{cases}$$

Is g submodular? (Give a proof or counterexample.)

(c) Can we find in polynomial time a subset $U \subseteq V$ that minimizes f? (Give a brief explanation.)

(d) Can we find in polynomial time a subset $U \subseteq V$ that maximizes g? (Give a brief explanation.)

Answer to Problem 1

(a) The function f is submodular:

- Let $A, B \subseteq V$ and $b := \max\{A \cup B\} \in B \setminus A$. Then

$$f(A \cup B) + f(A \cap B) = b + f(A \cap B) \leq b + f(A) = f(B) + f(A)$$

since $f(A \cap B) = 0$ for $A \cap B = \emptyset$ and otherwise since $A \cap B \subseteq A$ (and thus $\max\{h(a) : a \in A \cap B\} \leq \max\{h(a) : a \in A\}$).

- The case $a := \max\{A \cup B\} \in A \setminus B$ follows analogously.
- Let $A, B \subseteq V$ and $c := \max\{A \cup B\} \in A \cap B$. Then

$$f(A \cup B) + f(A \cap B) = c + c = f(A) + f(B).$$

(b) The function g is *not* submodular. Let $A := \{0\}$, $B := \{1\}$, and $N := 2$. Then

$$g(A \cup B) + g(A \cap B) = 1 + 1 = 2 \not\leq g(A) + g(B) = 0 + 1 = 1.$$

(c) Yes, because f is submodular (see (a)), and we can minimize submodular functions in polynomial time (see lectures). An even easier explanation: The minimizer is, of course, $f(\emptyset) = 0$ (because h maps into $\{0, \ldots, N\}$). This solution is found in $O(1)$.

(d) Yes. The maximum is clearly $\max\{g(\emptyset), g(V)\} = \max\{1, g(V)\}$. This solution is found in $O(1)$.

Problem 2
Give an example of a network with integral arc capacities that shows that the generic augmenting path algorithm for the maximum flow problem can have exponential running time.

Answer to Problem 2
Figure 1 shows an $s-t$ network with arc capacities 1 and C (a large integer).

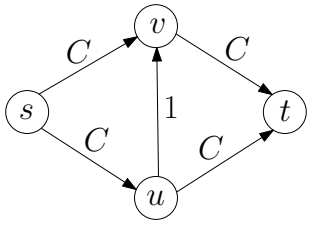

Figure 1:

A generic augmenting path algorithm might alternate between pushing 1 unit of flow along the augmenting path $s \to u \to v \to t$ and then pushing 1 unit of flow along the augmenting path $s \to v \to u \to t$. This requires augmentation along $2C$ augmention paths. Since we need $O(\log C)$ bits to store C, we have here an exponential running time.

Problem 3
Suppose you need to take n final exams. If you study h hours on Course $\#j$, $j = 1, \ldots, n$, you can obtain $p(h, j)$ grade points for that course. Suppose further that you are willing to study H hours overall, and you want to know the maximum number of grade points that you can get.

(a) Give a dynamic programming method for solving this problem. The total number of operations in your algorithm should be polynomial in n and H.

(b) What is the time complexity of your algorithm in (a)?

Answer to Problem 3

(a) Let $f(h, j)$ denote the *maximum* number of grade points that you can get by studying h in total split over the *first* j courses. We are interested in computing $f(H, n)$.

We have the following recursion:

$$\begin{aligned} f(h,1) &= p(h,1), && (h = 0, \ldots, H), \\ f(h,j) &= \max_{0 \le i \le h} \{f(h-i, j-1) + p(i,j)\} && (j \ge 2;\ h = 0, \ldots, H). \end{aligned}$$

(b) We need to fill in an $n \times H$ table. The number of operations to determine an entry is at most $2H$. The total number of operations is thus $O(nH^2)$.

Problem 4
In an instance of the Bin Packing Problem, n items of different volume $v_i \in (0, 1]$ must be packed into a finite number of bins, each of volume 1, such that the total number of bins used is minimal.

Consider the following algorithm for the Bin Packing Problem, which looks at the items in some arbitrary order and attempts to pack the next item into the first opened bin where it fits. If no bin is found, it opens a new bin and places the item in the new bin.

For example, suppose we have three items with $v_1 = 0.3$, $v_2 = 0.8$, and $v_3 = 0.2$. The first item will go to the first bin. The second item goes to the second bin since the first does not have enough room. The last item is then assigned to the first bin.

Show that this algorithm has an approximation guarantee of 2.

Answer to Problem 4

As the volume of each bin is 1, we have clearly

$$\sum_{i=1}^{n} v_i \leq \mathrm{OPT} \tag{1}$$

for the value OPT of the optimal solution. Let ALG denote the number of bins used in the algorithm. The algorithm cannot leave two bins less than half full (because the second of the two bins would not have been opened as all items fit into the first bin). Thus, in the solution of the algorithm there are at least $\mathrm{ALG} - 1$ bins that are more than half full. In other words, we have

$$\frac{1}{2}(\mathrm{ALG} - 1) < \sum_{i=1}^{n} v_i. \tag{2}$$

Combining (1) and (2) we have $\mathrm{ALG} - 1 < 2 \cdot \mathrm{OPT}$, and since ALG is integer we obtain

$$\mathrm{ALG} - 1 < \mathrm{ALG} \leq 2 \cdot \mathrm{OPT}.$$