Albert-Ludwigs-Universität, Institut für Informatik
Prof. Dr. Fabian Kuhn
M. Ahmadi, P. Schneider, J. Uitto

February 16, 2018

Algorithm Theory - Winter Term 2017/2018
Exercise Sheet 7

Hand in by Monday 14:15, February 5, 2018

The points on this exercise sheet are bonus points. Earned points are added to your score but this sheet will not be considered in the maximum number of achievable points. Note the earlier deadline!

Exercise 1: Maximum Coverage (10+3 Points)

Let X be a set of n elements and let $\mathcal{S} = \{S_1, \dots, S_m\}$ be a system of m subsets $S_1, \dots, S_m \subseteq X$. For an integer parameter $k \geq 1$, the maximum coverage problem asks for k sets $A_1, \dots, A_k \in \mathcal{S}$ such that $|\bigcup_{j=1}^{k} A_j|$ is maximized.

The following greedy approximation algorithm starts with $A = \emptyset$ and does k iterations. In iteration j it adds a subset $A_j \in \mathcal{S}$ to A (i.e. $A \leftarrow A \cup A_j$) that *maximizes* $|A \cup A_j|$ in the *current step*.

Let $O := \bigcup_{j=1}^{k} O_j$ with $O_1, \dots, O_k \in \mathcal{S}$ such that $|O|$ is *maximized overall*. Let $A^{(i)} = \bigcup_{j=1}^{i} A_j$, be the union of subsets chosen by the greedy algorithm until iteration i (i.e. $A^{(k)} = A$).

(a) Show that for any $i \in \{1, \dots, k\}$ it holds that $|O| - |A^{(i)}| \leq (1 - 1/k)^i |O|$.

(b) Show that the approximation ratio of the greedy algorithm is larger than $1 - \frac{1}{e}$.[1]

Sample Solution

(a) Consider the set $O \setminus A^{(i)}$. This set is covered by $O_1, \dots, O_k$. Therefore there must be a set O_j such that $O_j \setminus A^{(i)}$ covers more of the set $O \setminus A^{(i)}$ than $O_1 \setminus A^{(i)}, \dots, O_k \setminus A^{(i)}$ cover *on average*. That is

$$|O_j \setminus A^{(i)}| \geq |O \setminus A^{(i)}|/k \tag{1}$$

Assume our algorithm is in iteration $i+1$ and we choose the set $A_{i+1} \in S$ that maximizes the number of elements we add to $A^{(i)}$, i.e., we maximize $|A_{i+1} \setminus A^{(i)}|$. This greedy choice ensures that for any O_j, $j \in \{1, \dots, k\}$ we have

$$|A_{i+1} \setminus A^{(i)}| \geq |O_j \setminus A^{(i)}| \tag{2}$$

Putting Inequalities (1),(2) together we get

$$|A_{i+1} \setminus A^{(i)}| \geq |O_j \setminus A^{(i)}| \geq \frac{|O \setminus A^{(i)}|}{k} \geq \frac{|O| - |A^{(i)}|}{k} \tag{3}$$

And then

$$\begin{aligned} |O| - |A^{(i+1)}| &= |O| - |A^{(i)}| - |A_{i+1} \setminus A^{(i)}| \\ &\overset{(3)}{\leq} |O| - |A^{(i)}| - \frac{|O| - |A^{(i)}|}{k} = \left(1 - \frac{1}{k}\right)\left(|O| - |A^{(i)}|\right) \end{aligned} \tag{4}$$

[1] The constant e is Euler's number.

The rest can be done with induction. In the *base case* we have $A^{(1)} = A_1$ and $A^{(0)} = \emptyset$ hence

$$|O| - |A^{(1)}| \overset{(4)}{\leq} (1 - \frac{1}{k})|O|.$$

Presume the claim is true for iteration i of the algorithm. We show the claim for $i+1$:

$$|O| - |A^{(i+1)}| \overset{(4)}{\leq} \left(1 - \frac{1}{k}\right)\left(|O| - |A^{(i)}|\right) \overset{\text{ind.-hyp.}}{\leq} \left(1 - \frac{1}{k}\right)^{i+1}|O|.$$

(b) We showed that $|A^{(i)}| \geq \left(1 - (1 - 1/k)^i\right)|O|$ is true for all i. Then, for $i = k$, it holds

$$\begin{aligned} |A| &= |A^{(k)}| \\ &\geq \left(1 - (1 - 1/k)^k\right)|O| \\ &\geq (1 - 1/e)|O| \end{aligned}$$

Exercise 2: Acyclic Graphs (10 Points)

Consider the following problem. Given a directed graph $G = (V, E)$, the goal is to determine a maximum cardinality set $E' \subseteq E$ such that the graph induced by E' is acyclic. Provide a $\frac{1}{2}$-approximation algorithm for this problem. Prove that the approximation ratio is at least $\frac{1}{2}$.

Hint: Considering any set of nodes arbitrarily partitioned into two sets A and B, the set of outgoing edges from A to B induce an acyclic graph as well as the set of outgoing edges from B to A.

Sample Solution

Algorithm: Our algorithm `makeAcyclic` with input $G = (V, E)$ works recursively as follows. In the base case we have a single node graph which is acyclic and can be returned as is. Otherwise it partitions V into two sets $A \cup B = V$ (for performance reasons of (almost) equal size (± 1)). Then it determines the set $E_{A\to B} = \{(u, w) \in E \mid u \in A, w \in B\}$ of edges from A to B and analogously $E_{B\to A}$. Let $E_{\max}$ be the larger set of $E_{A\to B}$ and $E_{B\to A}$. Via recursions of this algorithm on the graphs $G_A = (A, E_A)$ and $G_B = (B, E_B)$ induced by A and B respectively, we determine acyclic graphs $G'_A = (A, E'_A)$ and $G'_B = (B, E'_B)$. Our output is the graph $(V, E_{\max} \cup E'_A \cup E'_B)$.

Algorithm 1: `makeAcyclic`$(G = (V, E))$

if $|V| = 1$ **then**
  return G
Partition V into subsets $A \cup B = V$ of similar size
$E_{A\to B} \leftarrow \{(u, w) \in E \mid u \in A, w \in B\}$
$E_{B\to A} \leftarrow \{(u, w) \in E \mid u \in B, w \in A\}$
$E_{\max} \leftarrow$ larger of $E_{A\to B}, E_{B\to A}$
$G_A = (A, E_A), G_B = (B, E_B) \leftarrow$ graphs induced by A, B
$G'_A = (A, E'_A) \leftarrow$ `makeAcyclic`(G_A)
$G'_B = (B, E'_B) \leftarrow$ `makeAcyclic`(G_B)
return $(V, E' = E_{\max} \cup E'_A \cup E'_B)$

Correctness: As the runtime was not an issue, we only have prove that the above algorithm gives us a $\frac{1}{2}$-approximation. We prove this by showing that the resulting graph contains at least half of all edges. Since an optimal solution can have at most $|E|$ edges, this obviously proves the claim.
We make an *induction* over the number of nodes of the input graph. In the *base case* $|V| = 1$ we simply return the single node graph with no edges which for trivial reasons is acyclic and contains half of this graphs edges ($\frac{0}{2} = 0$).

For the *induction step* Let $G = (V, E)$ with $|V| > 1$. Our *hypothesis* is that for all subgraphs with *strictly fewer* than $|V|$ nodes our algorithm computes an acyclic graph with at least half the original number of edges in that subgraph. With this hypothesis we immediately obtain that E'_A and E'_B contain at least half the edges of E_A and E_B respectively. It remains to be shown that $E_{\max}$ contains at least half of the remaining edges $E \setminus (E_A \cup E_B)$. Since $E_{\max}$ is the larger set of the partition $E_{A\to B} \cup E_{B\to A} = E \setminus (E_A \cup E_B)$ this is also true.

Exercise 3: Online Vertex Cover (9+4+4 Points)

Let $G = (V, E)$ be a graph. A set $S \subseteq V$ is called a *vertex cover* if and only if for every edge $\{u, v\} \in E$ at least one of its endpoints is in S. The minimum vertex cover problem is to find such a set S of minimum size.
We are considering the following online version of the minimum vertex cover problem. Initially, we are given the set of nodes V and an empty vertex cover $S = \emptyset$. Then, the edges appear one-by-one in an online fashion. When a new edge $\{u, v\}$ appears, the algorithm needs to guarantee that the edge is covered (i.e., if this is not already the case, at least one of the two nodes u and v needs to be added to S). Once a node is in S it cannot be removed from S.

(a) Provide a deterministic online algorithm with competitive ratio at most 2. That is, your online algorithm needs to guarantee at all times that the vertex cover S is at most by a factor 2 larger than a current optimal vertex cover. Prove the correctness of your algorithm.

(b) Show that any deterministic online algorithm for the online vertex cover problem has competitive ratio at least 2.

(c) Use Yao's principle to show that any randomized online algorithm for the online vertex cover problem has competitive ratio at least 3/2.

Sample Solution

(a) **Online Algorithm:** Start with $S = \emptyset$. When a new edge appears our online algorithm first checks whether the new edge is already covered, i.e., if it has at least one endpoint in S. If that edge is not covered the algorithm adds *both* endpoints of the new edge to S; if the edge is already covered then the algorithm adds no point to S.

Correctness: Our online algorithm guarantees that any edge that appears will have at least one endpoint in S. Therefore, the set S is a vertex cover of the graph.

Claim: The above online algorithm provides a competitive ratio of 2.

Proof. Let M be the set of those edges $\{u, v\}$ with $u \in S$ and $v \in S$. We have $|M| = \frac{|S|}{2}$.

We have seen in the lecture that the size of a matching of a graph is upper bounded by the size of a vertex cover of the graph.[2] Let S^* be the smallest possible vertex cover. If we could show that M is a matching then we would have $\frac{|S|}{2} = |M| \leq |S^*|$. Hence the claim holds if we show that M is a matching. Our online algorithm guarantees that for any two edges $\{u, v\}$ and $\{u', v'\}$ in M, the edges are not incident.

More precisely, assume both endpoints of $\{u, v\}$ are added to S. Therefore the edge is added to M. There can be no edges in M which are adjacent $\{u, v\}$ and were added to M before $\{u, v\}$, otherwise $\{u, v\}$ would not have been added to M in the first place. Let $\{u', v'\}$ be an edge that appears after $\{u, v\}$. It will be added to M only if $u \neq u'$ and $v \neq v'$. This guarantees that these two edges in M are not incident. Hence M is a matching. □

[2] Fix a vertex cover and a matching of a graph. Then any edge of the matching has an endpoint in the vertex cover (every edge is covered). On the other hand every node in the vertex cover has at most one adjacent edge in the matching (lest the condition of a matching would be violated).

(b) Let us fix OPT to be an optimal offline algorithm and ALG to be any online algorithm. We (as an adversary) construct a scenario with only two incident edges where ALG has to put two vertices in S while OPT can cover both edges by only one vertex.

We send the first edge. If ALG puts both endpoints of the first edge in S then the second edge can be connected to any endpoint of the first edge. If ALG adds only one of the endpoints of the first edge to S then we connect the second edge to the endpoint of the first edge that is not in S. So the second edge has already no endpoint in S and ALG has to add at least one of the endpoint of the second edge to S. Therefore ALG returns S whose size is at least 2.

By contrast, OPT chooses only the middle vertex to cover both edges. Note that OPT can do this since it is offline and knows the input sequence in advance. This means that for any deterministic algorithm ALG we have a online order of edges of a graph G (given above), such that for output S of ALG and the minimum vertex cover S^* of G we have $|S| \geq 2|S^*|$. This proves that we have a *strict* competetive ratio of at least 2 for deterministic algorithms.

Remark: For this exercise we were satisfied with a scenario showing that any deterministic algorithm has a strict competitive ratio of at least 2.

In order to prove the same for a competitive ratio of 2 (without the keyword strict) we have to do more. For a contradiction assume that we could do better than 2-competitive, i.e., for constants $\alpha \geq 0$ and $1 \leq c < 2$ we have $|S| \leq c \cdot |S^*| + \alpha$.

We duplicate the above scenario k times, meaning that we have k pairs of edges which we send successively. Like before, we always connect each pair of edges in such a manner that a given deterministic algorithm ALG needs two nodes for each pair to cover both edges, while OPT needs just one (the shared middle node). This means that $|S| = 2k$ and $|S^*| = k$. Then $|S| \leq c \cdot |S^*| + \alpha$ is equivalent to $k \leq \frac{\alpha}{2-c}$. The contradiction arises from the fact that the size k of our scenario can be made arbitrarily large.

(c) We construct a randomized input and show that the *best deterministic algorithm for that kind of input distribution* has an expected (strict) competitive ratio of $\frac{3}{2}$. The basic idea is the same as in part (b). As before, we send two adjacent edges. Let $\{u, v\}$ be the first edge being sent. The second edge is subject to a random choice. With probability $\frac{1}{2}$ we attach it to u otherwise to v.

As before ALG has to choose after the first edge $\{u, v\}$ which node(s) it adds to S. Since ALG does not know whether the next edge will attach to u or to v, the smartest thing ALG can do is to just pick one node, w.l.o.g. u and hope that the next edge will attach to it.

Note that ALG needs to add at least one endpoint of $\{u, v\}$ to S since by problem formulation every edge needs to be covered immediately. Also note that it would be stupid for ALG to add both endpoints of $\{u, v\}$ to S right away, since then it would definitely require two nodes even though it could gamble and hope to use just one.

So w.l.o.g. ALG adds u to S and with probability $\frac{1}{2}$ it wins and the next edge attaches to u and it needs only one node to cover both edges. However, with probability also $\frac{1}{2}$ ALG looses and the next edge attaches to v, thus ALG has to add a second node to S to produce a vertex cover.

The expectation is $\mathbb{E}(|S|) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2 = \frac{3}{2}$. As Before OPT still requires only one edge. By Yao's principle the expectation of the best randomized algorithm on a worst case input can only be as good as the best deterministic algorithm on a worst case randomized input. Thus any randomized algorithm for the online vertex cover problem has a *strict* competitive ratio at least $\frac{3}{2}$.

Remark: Again we were satisfied with the proof that a randomized algorithm has a strict competitive ratio of at least $\frac{3}{2}$. For the proof that there is no randomized algorithm better than $\frac{3}{2}$-competitive (without the keyword strict) we can employ the same proof by contradiction as in (b).