University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger

Algorithm Theory
Sample Solution Exercise Sheet 8

Exercise 1: Online Vertex Cover *(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.[1] 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 $\{u, v\}$,

[1] 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).

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. □

(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).

Exercise 2: Ski Rental Problem *(Points)*

Assume you go skiing and have to decide whether to buy or rent skis. Assume that cost is your only concern and that buying ski is k times more expensive than renting it for a day (assume that k is an integer).
After buying your own ski you can use them forever without additional cost. Finally assume that you have no knowledge how many days n you will ever go skiing in total.
This means that as long as you go skiing you have to decide on every new skiing-day in an online fashion whether to buy or rent, until you finally decide to buy a pair of ski.

(a) Describe the best offline strategy OPT (n is known beforehand) and give the cost as a function depending on n.

(b) Assume your online strategy ALG (where you have no knowledge of n) is to buy a pair of ski before your first skiing day. Give an upper bound for the strict competitive ratio of ALG and explain it briefly.

(c) Give a strategy ALG' that is strictly 2-competitive and prove it.

Sample Solution

(a) If $n \geq k$, buy skis on the first day, otherwise rent it for n days. If x are the costs for one day, the total costs are $\mathrm{OPT}(n) = x \cdot \min\{k, n\}$.

(b) If you buy skis on the first day, you pay $\mathrm{ALG} = x \cdot k$. We have $\mathrm{OPT} \geq x$ and thus $\mathrm{ALG}/\mathrm{OPT} \leq k$.

(c) The online strategy ALG' works as follows: Rent skis on each day $\leq k$ and buy skis on day $k+1$. For $n \leq k$ we have ALG' $= xn = \mathrm{OPT}$ and for $n > k$ we obtain ALG' $= 2xk$ and $\mathrm{OPT} = xk$ and hence $\mathrm{ALG}/\mathrm{OPT} = 2$.

Exercise 3: Maximum Cut *(Points)*

Let $G = (V, E)$ be an unweighted undirected graph. A maximum cut of G is a cut whose size is at least the size of any other cut in G.

(a) Give a simple randomized algorithm that returns a cut of size at least $1/2$ times the size of a maximum cut *in expectation* and prove this property.

(b) Prove that the following deterministic algorithm (Algorithm 1) returns a cut of size at least $1/2$ times the size of a maximum cut.

Algorithm 1 Deterministic Approximate Maximum Cut

```
Pick arbitrary nodes v_1, v_2 ∈ V
A ← {v_1}
B ← {v_2}
for v ∈ V \ {v_1, v_2} do
    if deg_A(v) > deg_B(v) then        ▷ deg_X(v) is the number of v's neighbors in X ⊆ V.
        B ← B ∪ {v}
    else
        A ← A ∪ {v}
Output A and B
```

(c) Let us now consider an online version of the maximum cut problem, where the nodes V of a graph $G = (V, E)$ arrive in an online fashion. The algorithm should partition the nodes V into two sets A and B such that the cut induced by this partition is as large as possible. Whenever a

new node $v \in V$ arrives together with the edges to the already present nodes, an online algorithm has to assign v to either A or B. Based on the above deterministic algorithm (Alg. 1), describe a deterministic online maximum cut algorithm with *strict competitive ratio* at least 1/2. You can use the fact that Algorithm 1 computes a cut of size at least half the size of a maximum cut.

Hint: An online algorithm for a maximization problem is said to have strict competitive ratio α if it guarantees that $\text{ALG} \geq \alpha \cdot \text{OPT}$*, where* ALG *and* OPT *are the solutions of the online algorithm and of an optimal offline algorithm, respectively.*

(d) Show that no deterministic online algorithm for the online maximum cut problem can have a strict competitive ratio that is better than 1/2.

Sample Solution

(a) Initialize $S = \emptyset$. Each node joins S independently with probability 1/2. For an edge $e = \{u, v\}$, the probability that e is between S and $V \setminus S$ is $\Pr(u \in S \wedge v \notin S) + \Pr(u \notin S \wedge v \in S) = 1/2$. So in expectation, half of the edges are between S and $V \setminus S$.

(b) We distinguish between *crossing edges* with one endpoint in A and one in B and *inner edges* with either both endpoints in A or both in B. We show that the following property is a loop-invariant: The number of crossing edges is at least the number of inner edges.

This is true before entering the loop the first time (there are no inner edges). In the following iterations, a node is added to B iff it has more neighbors in A than it has in B and it is added to A iff it has at least as many neighbors in B as in A. So in either case, the number of crossing edges that are added is at least the number of inner edges that are added.

When all nodes are processed, the number of crossing edges is at least the number of inner edges, i.e., at least half of the edges are crossing edges. So the resulting cut has size at least $|E|/2$ and $|E|$ is an upper bound for a maximum cut.

(c) Algorithm 1 actually describes an online algorithm. The first node that arrives is assigned to A, the second node to B and all other nodes are processed as described in the loop. As shown we obtain $\text{ALG} \geq |E|/2$ and $\text{OPT} \leq |E|$ and hence the competitive ration is at least 1/2.

(d) Consider a graph with nodes $v_1, v_2, \ldots, v_n$. Nodes v_1 and v_2 are adjacent to each other node, more edges do not exist. The cut $(\{v_1, v_2\}, \{v_3, \ldots, v_n\})$ has size $2(n-2) \leq \text{OPT}$. Assume ALG is an online algorithm with competitive ratio $r > 1/2$ which computes a cut $(S, V \setminus S)$ for any graph (V, E). If ALG first receives v_1 and v_2, it must put one in S and the other in $V \setminus S$, because otherwise we had $\text{ALG} = 0$ and $\text{OPT} = 1$ in case the graph consists only of v_1 and v_2 (which an online algorithm can not know at this point). Any cut with $v_1 \in S$ and $v_2 \notin S$ (or vice versa) has size $n - 1$. Therefore we have $\text{ALG}/\text{OPT} \leq \frac{n-1}{2(n-2)} \overset{n \to \infty}{\longrightarrow} 1/2$. So for n sufficiently large we have $\text{ALG}/\text{OPT} < r$, contradicting that ALG has competitive ratio r.