Collab w/
* Luke Fraker
+ Pankaj Parag

CSE 521p: Applied Algorithms HW1

Anthony Adkins (anadkins@uw.edu)

* 1. To prove that every deterministic online algorithm has a competitive ratio of at least $2 - \frac{2}{n+1}$,

① I will show that there exists a request ~~[illegible]~~ sequence σ for T of n items ~~that~~ has a ~~[illegible]~~ cost of $T \cdot n$

② And that for such a sequence, there is an optimal offline algorithm with a cost of $\frac{1}{2}T(n+1) + C$ for some constant C independent of T

① and ② are sufficient proof because for a competitive ratio:

$$\forall\sigma : \text{online} = \text{C.R.} \cdot \text{optimal offline} \leq \left(2 - \frac{2}{n+1}\right)\left(\frac{1}{2}T(n+1) + C\right)$$

$$= T(n+1) + 2C - T - \frac{2C}{n+1} = \underbrace{T \cdot n}_{\text{online cost}} + \frac{2Cn}{n+1}$$

To prove ①, we say sequence σ requests ~~an~~ element n (the last element of the list) regardless of any swaps of items or moves made by the online algorithm. Thus, every request costs n, so for sequence of length T, we have a cost of $n \cdot T$.

To prove ②, we first show that the worst-case offline performance is on a sequence where no item is requested more than once ($T \leq n$)

If $T = n$ and no items are repeated, the cost is

$$n + (n-1) + (n-2) + \ldots + 2 + 1 = \sum_{k=1}^{n} k = \frac{1}{2}(n+1)n = \frac{1}{2}T(n+1)$$

If $T < n$ and no items are repeated, we can first swap so that all requested items precede all non-requested items in the list.

Then, the cost is $T + (T-1) + \ldots + 1 = \frac{1}{2}T(T+1) + C$
where C is the cost of the initial rearrangement.
Of course, there exists a list where $C = \emptyset$, where all request items are already at the beginning of the list, for which the cost is simply
$\frac{1}{2}T(T+1) \leq \frac{1}{2}T(n+1)$ because $T \leq n$

Finally, because C depends on n, and not T,
$C \rightarrow \emptyset$ as $T \rightarrow \infty$

2a). To prove Swap Forward has a competitive ratio of $\Omega(n)$, we will give an example sequence σ such that $SF(\sigma) \geq c \cdot n \cdot OPT(\sigma)$ for some constant c

Consider a linked list of n items where $n > 2$

$$\underset{1}{O} \to \underset{2}{O} \to \cdots \to \underset{n-1}{O} \to \underset{n}{O}$$

the request sequence consists of alternating requests for items n and $n-1$ (continuously the last item in the list).

We also say $|\sigma| \geq n$, so that the number of requests made exceeds n

$SF(\sigma)$ costs n for each request, and there are $\geq n$ requests, so that the cost of $SF(\sigma) \geq n^2$

The optimal algorithm retrieves n, moves it to the front, then retrieves $n-1$ and moves it second to the front: $\underset{n}{O} \to \underset{n-1}{O} \to \underset{1}{O} \to \underset{2}{O} \to \cdots \to \underset{n-2}{O}$

Then, all subsequent requests cost 1 or 2

So, $OPT(\sigma) = n + n + \frac{1}{2}(|\sigma| - 2) + (|\sigma| - 2)$

$= 2n + \frac{3}{2}|\sigma| - 3$

Now, we can say

$SF(\sigma) \geq n \cdot |\sigma| \geq c \cdot n(2n + \frac{3}{2}|\sigma| - 3)$

$\Rightarrow SF(\sigma) \geq 2cn^2 + (\frac{3c}{2}n)|\sigma| - 3cn$

for a sufficiently small c where $|\sigma| \geq n$

where $c < \frac{2}{3}$, but approaches $\frac{2}{3}$ as $|\sigma| \to \infty$ with a fixed n, 2 b). For the online counter (OC) algorithm, consider a sequence of requests σ for a list of n items $a_1, \ldots, a_n$ where the first $2n$ accesses are for item a_1, the next $2n-1$ for a_2, the next $2n-2$ for a_3, until the final $n+1$ for item a_n

The cost of $OC(\sigma) = 2n + 2(2n-1) + 3(2n-2) + \ldots + n(n+1)$ because the list would never be rearranged based on its counters ($c_1(t) > c_2(t) > \ldots c_n(t)$ for all t).

To simplify the cost, $OC(\sigma) = \sum_{i=1}^{n} i(2n+1-i)$

The optimal algorithm would retrieve a_1 n times at a cost of 1, then retrieve a_2 once at a cost of 2 ~~and~~ move it to the front, serving the remaining $2n-2$ requests at a cost of 1, continuing this way until retrieving item a_n for cost n and serving the remaining n requests for a_n at a cost of 1 each

Thus, $OPT(\sigma) = 2n + (2+2n-2) + (3+2n-3) + \ldots + (n+n)$

$$= n \cdot 2 \cdot n = 2n^2$$

Whereas $OC(\sigma) = \sum_{i=1}^{n} i(2n+1-i) = n(n+1)n - \left(\frac{(n-1)^3}{3} + (n-1)^2 + \frac{2(n-1)}{3}\right)$

$$= n^3 + n^2 - \frac{1}{3}(n^3 - 3n^2 + 3n - 1) - (n^2 - 2n + 1) - \frac{2}{3}n + \frac{2}{3}$$

$$= \frac{2}{3}n^3 + n^2 + \frac{n}{3}$$

So, to show $OC(\sigma) = \Omega(n)$, we say

$$\frac{2}{3}n^3 + n^2 + \frac{n}{3} \geq c \cdot n \cdot 2n^2 = c \cdot 2n^3$$

for a sufficiently small c

because the ~~#~~ n factor dominates the cost as it increases as both sides are cubic.

Note that if we repeat this sequence σ, c needs not be quite as small.

In fact, $c < \frac{1}{3}$ clearly, but $c \to \frac{1}{3}$ as either $n \to \infty$ or $|\sigma| \to \infty$ (repeating the request sequence)

4. Given a set of buyers B, items I, connected by a bipartite graph. We will show that a greedy algorithm guarantees at least half the optimal total profit.

That is, for S_i items assigned to buyer i with budget B_i, the profit is $\min(B_i, \sum_{j \in S_i} v_{ij})$ for buyer i where v_{ij} is buyer i's value for item j.

For our algorithm, we take each item individually, and assign it to the buyer with maximum utility, which we define to be the remaining budget the buyer has for that item, or:

$\cancel{\text{utility}_{ij} = \min(v_{ij}, B_i - \sum_{k \in B_i} v_{ik})}$ ~~where M_i is the set of items already assigned to buyer i.~~

$$\text{utility}_{ij} = \min(v_{ij}, B_i - \sum_{k \in S_i} v_{ik})$$ where S_i is the set of items already assigned to buyer i

This way, we maximize the largest additional profit as each item is assigned.

For this online algorithm, our profit is $\sum_i \min(B_i, \sum_{j \in S_i} v_{ij})$

Now consider the case of two buyers and item j

For this first assignment of j, our possible online profit is

Greedy(j)

Say that buyer 1 has greater utility for item j, so item j is assigned to S_1 $(\min(v_{1j}, B_1) > \min(v_{2j}, B_2))$

Then, $\text{Greedy}(j) = \min(v_{1j}, B_1)$

and $\text{OPT}(j) \leq \min(v_{1j}, B_1) + \min(v_{2j}, B_2)$

Because Greedy preferred buyer 1 because of their utility, We can then say that

$$\text{Greedy}(j) \geq \tfrac{1}{2}\,\text{OPT}(j)$$

We can then proceed to item k with modified budgets $B'_1 := \max(B_1 - v_{1j}, 0)$, $B'_2 := \max(B_2 - v_{2j}, 0)$ and $B'_i := B_i$ for all $i \geq 3$

We can say $\text{OPT}(j) \leq \min(v_{1j}, B_1) + \min(v_{2j}, B_2) + \text{OPT}(L)$

where $\text{OPT}(L)$ is the remaining profit from items $> j$, with modified budgets B'.

Now, say for item k, the online algorithm sees $\min(B'_1, v_{1k}) > \min(B_2, v_{2k})$ and assigns k to buyer 1

Now, $\text{Greedy}(j,k) = \min(v_{1j}, B_1) + \min(v_{1k}, B'_1)$

$\text{OPT}(j,k) \leq \min(v_{1j}, B_1) + \min(v_{2j}, B_2) + \min(v_{1k}, B'_1) + \min(v_{2k}, B_2$

~~Because $B_2 - v_{2k} < B'_1 - v_{1k}$ and $\min(v_{2j},$~~

Because $\min(B'_1, v_{1k}) \geq \min(B_2, v_{2k})$ and $\min(B_1, v_{1j}) \geq \min(B_2, v_{2j})$

$$\text{Greedy}(j,k) \geq \tfrac{1}{2}\,\text{OPT}(j,k)$$

Thus, we can see that for each additional assignment, the optimal offline is bound by the remaining budgets plus the possible value gained by assigning the item to both buyers

So that $\text{OPT}(L)$ will always be $\leq 2 \cdot \text{Greedy}(L)$

Or, Greedy is 2-competitive

5. a). A 2-competitive online algorithm is For each buyer i, pick the first item in their set of items S_i and assign that item to buyer i. If all items in S_i are already assigned, this buyer does not get an item.

~~I will prove this is 2-competitive because all deterministic algorithms are 2-competitive in (b)~~

This algorithm is 2-competitive because

(i) there exists a sequence for which it performs twice as bad as optimal

Consider buyer 1 with $S_1 = \{1, 2\}$
and buyer 2 with $S_2 = \{1\}$

The algorithm assigns item 1 to buyer 1, and buyer 2 is not assigned.

Optimally, buyer 1 gets item 2 and buyer 2 gets item 1

So the algorithm is at best 2-competitive

ii) This is the worst case scenario, because if there exists an optimal algorithm that achieves k matches, there are k possible matches

Thus, assigning 1 item can at most eliminate all possible items for 1 other buyer.

If assigning 1 item eliminated multiple buyers, it could not have been true that k matches existed.

because that would imply that two other buyers both only wanted that 1 item.

Thus, in the worst case, each item assigned can only eliminate one other buyer from getting an item in their set S_i.

Following this, if k possible ~~items~~ buyers could have been matched, at least half of the matches will be achieved ($\frac{1}{2}k$ of optimal k).

And so, the algorithm achieves a 2-competitive ratio.

b.) This proof that any deterministic algorithm has at least a competitive ratio of 2 requires the example where there are 2 buyers and 2 items.

Buyer 1 has both items in S_1

Buyer 2 only has whichever item the online algorithm assigns to Buyer 1 in their S_2 set

Thus, online can only assign one item, but optimally an offline assigns two

Therefore any deterministic algorithm has a competitive ratio of at least 2.

(6) For algorithm Rand that randomly assigns each buyer i an item from their desired set S_i

we will show that it is $2 - o(1)$ competitive by showing that worst case it achieves a performance

$$\text{Rand} \geq \tfrac{1}{2}\text{OPT} - o(1)$$

where the term $o(1)$ is derived from the fact that the probability of this worst-case scenario being achieved approaches $\emptyset$ as $n \to \infty$ or,

$$\text{Probability}\left(\text{Rand} = \tfrac{1}{2}\text{OPT}\right) \to \emptyset \quad \text{as } n \to \infty$$

Consider the case of $2n$ buyers and $2n$ items, where Buyer i is equally happy to get any items $i, n+1, n+2, \ldots 2n$ while $i \leq n$

and Buyer i only wants i when $i > n$

That is, $S_i = \{i, n+1, n+2, \ldots, 2n\}$ when $i \leq n$

and $S_i = \{i\}$ when $i > n$

Our algorithm Rand can clearly achieve $2n$ assignments, if by chance it assigns buyer i to item i

This is also the optimal offline assignment such that all $2n$ buyers and $2n$ items are matched.

However, worst-case Rand will not assign any of the first n items to their respective buyers i.

In this case, all items $n+1$ to $2n$ are assigned to buyers 1 through n, leaving no further matches.

Then, only n matches are achieved, so Rand only achieves $\tfrac{1}{2}\text{OPT}$.

~~so~~ However, if Rand assigns any of the first n items to its respective buyer i, then ~~Rand~~

$$\text{Rand} > \frac{1}{2}\text{OPT}.$$

The probability of this worst case where $\text{Rand} = \frac{1}{2}\text{OPT}$, can be calculated to be such that buyer 1 ~~picks~~ gets ~~any other items~~ any of items $n+1$ through $2n$ then buyer 2 gets any of the remaining of $n+1$ through $2n$, up to buyer n choosing randomly between ~~m~~ item n and the last item $>n$.

This probability can be calculated as:

$$\left(\frac{n}{n+1}\right) \times \left(\frac{n-1}{n}\right) \times \left(\frac{n-2}{n-1}\right) \ldots \left(\frac{2}{3}\right) \times \left(\frac{1}{2}\right) = \frac{1}{n+1}$$

Thus, $\text{Probability}(\text{Rand} = \frac{1}{2}\text{OPT}) = \frac{1}{n+1}$ in this case and so $\text{Probability}(\text{Rand} > \frac{1}{2}\text{OPT}) = \frac{n}{n+1}$

Therefore, the worst case probability $\frac{1}{n+1}$ ~~clearly~~ clearly goes to $\emptyset$ as n goes to ∞

Therefore, we can say that Rand has a competitive ratio of $2 - o(1)$