ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

WS 2005/06 Prof. R. Wattenhofer / Thomas Moscibroda / Stefan Schmid

Discrete Event Systems
Exercise 11: Sample Solution

1 Competitive Analysis

a) Recall that calls have infinite duration. Therefore, once a cell accepts a call, no neighboring cell can accept a call thereafter. The natural greedy algorithm $\mathcal{A}_{Greedy}$ accepts a call, whenever this is possible. That is, a call in cell C is accepted if no neighboring cell of C has previously accepted a call.

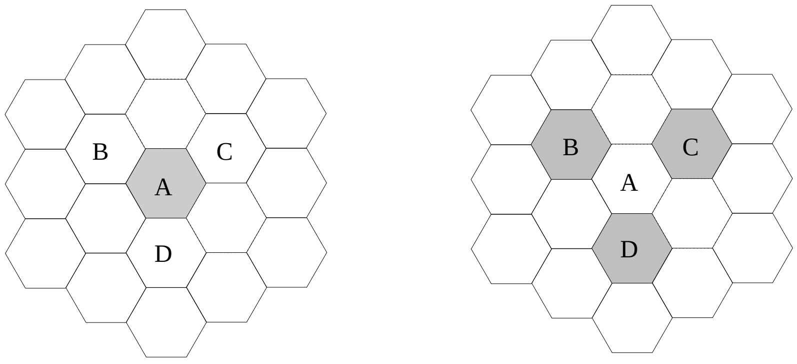

Figure 1: The solutions $\mathcal{A}_{Greedy}$ (left) and $\mathcal{A}_{Opt}$ (right)

By accepting a call, $\mathcal{A}_{Greedy}$ can prevent itself from accepting at most 3 subsequent calls. This is shown in Figure 1. Assume that there are four calls, the first one in A, then three non-interfering ones in neighboring cells B, C, and D. $\mathcal{A}_{Greedy}$ accepts the first and has benefit 1. $\mathcal{A}_{Opt}$ rejects the first call, but accepts the remaining three, resulting in a benefit of 3. The algorithm is 3-competitive.

b) There is no competitive algorithm if calls can have arbitrary durations. Assume that the first call arrives in A and has arbitrary duration. There are two possible actions for an algorithm ALG.

If ALG rejects this call, no further call will arrive and therefore $benefit(ALG) = 0$. The optimal algorithm would have accepted the call, i.e., $benefit(OPT) = 1$. The competitive ratio is infinitely large.

On the other hand, if ALG accepts the call, there will be infinitely many calls coming in state B, each of which has very short duration ϵ. While ALG cannot accept any of these calls (because the call in A has infinite duration), the optimal algorithm rejects the first call and accepts all subsequent calls. This yields $benefit(ALG) = 1$ as opposed to $benefit(OPT) = \beta$, for an arbitrarily large value of β

c) At first sight, it seems that there is no better algorithm than the natural greedy algorithm from part a) of the exercise. After all, the algorithm must accept the first call in order to stay competitive. Accepting the first call, on the other hand, leads to a competitive ratio of 3. However, it can be shown that there exists a *randomized algorithm* with competitive ratio 2.97. This algorithm accepts every call with a certain probability.

2 Online Algorithm

a) When using the strategy recommended by Cons-ULT, Mario has to serve all clients while Luigi does not move at all. The total length of Mario's path is $0.4+0.3+0.4+0.2+0.7 = 2$. In the optimal solution on the other hand, Luigi serves the first request even though he is more distant to this request than Mario. All subsequent requests are handled by the closer friend. Mario's path has length $0.1 + 0.1 = 0.2$ and Luigi walks a distance of $0.6 + 0.1 + 0.2 = 0.9$. Hence, the total cost of this solution is 1.1. For the request sequence σ_1, the competitive ratio of Cons-ULT's algorithm is therefore $2/1.1 \approx 1.82$.

b) No, the algorithm proposed by Cons-ULT is not competitive for any constant. In fact, the outcome of Cons-ULT's algorithm can be as bad as n times worse than the optimal solution, where n is the number of requests.

To see this, consider the request sequence $\sigma_2 = \frac{1}{2} - \epsilon\ ,\ 0\ ,\ \frac{1}{2} - \epsilon\ ,\ 0\ ,\ \frac{1}{2} - \epsilon\ ,\ \ldots$, for an arbitrarily small constant ϵ. Clearly, all requests of σ_2 are handled by Mario who has to go back and forth between the two points 0 and $\frac{1}{2} - \epsilon$. Hence, the total cost of Cons-ULT's algorithm is $n \cdot (\frac{1}{2} - \epsilon) \approx n/2$.

In contrast, the optimal solution for sequence σ_2 is much better. Luigi could move for the first request to position $\frac{1}{2} - \epsilon$, and the two friends could remain at their position forever thereafter. Hence, the optimal cost is $\frac{1}{2} + \epsilon$. Combining this with the result of Cons-ULT, we see that the competitive ratio α of the Cons-ULT ratio can be as bad as

$$\frac{n \cdot (\frac{1}{2} - \epsilon)}{\frac{1}{2} + \epsilon} \approx n.$$

c) Indeed, there exists a much better algorithm for Mario and Luigi's problem that achieves a competitive ratio of 2.[1] Let us assume that Mario's position is always to the left of Luigi or at exactly the same place. It can easily be seen that the algorithm does never change this invariant. Then, the algorithm handles the next request as follows:

- If the request is located to the left of Mario or to the right of Luigi, serve the request with the closer of the two.
- Otherwise, the request is between the two friends. In this case, both Mario *and* Luigi move towards the request at equal speed until (at least) one of them reaches the request.

We now prove that the above algorithm is 2-competitive.

Proof. Let OPT and ALG denote the optimal solution and the solution computed by our algorithm for a given request sequence, respectively. Also, let o_M^i and o_L^i be the positions of Mario and Luigi after the ith request in the optimal solution OPT. Similarly, we denote by a_M^i and a_L^i the positions of Mario and Luigi after the ith request in ALG.

[1]Note that in the exam, we did of course not expect students to write down a formal proof as detailed as in this master solution. Instead, it was sufficient to provide a reasonable argument why the proposed algorithm is competitive.

We now define the following potential function:

$$\Phi_i \;=\; 2(|o_M^i - a_M^i| + |o_L^i - a_L^i|) + |a_M^i - a_L^i|.$$

The potential function has three properties:

i) $\Phi_i \geq 0$ for all i.

ii) If OPT moves its players a distance d, then $\Phi_{i+1} \leq \Phi_i + 2d$.

iii) If ALG moves its players a distance d, then $\Phi_{i+1} \leq \Phi_i - d$.

Property i) certainly holds. As for property ii), notice that if OPT moves, the term $|a_M^i - a_L^i|$ is not changed and clearly, the term $|o_M^i - a_M^i| + |o_L^i - a_L^i|$ can increase by at most d. In order to prove that iii) holds, we assume that $o_M^i \leq o_L^i$, i.e., the Mario is never to the right of Luigi in the optimal solution. This is without loss of generality, because if the optimal solution switches the positions of the two friends, we can simply use a "renaming" in order to reestablish the invariant.

We must distinguish two cases. First, assume that the ith request is either to the left of Mario or to the right of Luigi, i.e., only one of the two moves by a distance of d. In this case, the term $|a_M^i - a_L^i|$ increases by d. On the other hand, the term $|o_M^i - a_M^i| + |o_L^i - a_L^i|$ must decrease by d, because after moving Mario or Luigi by a distance d, either $|o_M^i - a_M^i|$ or $|o_L^i - a_L^i|$ becomes 0. This is the case because the optimal solution must also handle this request. The new value of the potential function in the first case is therefore at most $\Phi_{i+1} \leq \Phi_i - 2d + d = \Phi_i - d$.

Now, consider the second case, in which the request is between Mario and Luigi and both of them move a distance $d/2$ towards the request from opposite sides. Because both friends move towards each other, the term $|a_M^i - a_L^i|$ must decrease by d. Now, consider the change of the term $|o_M^i - a_M^i| + |o_L^i - a_L^i|$. When serving the ith request, either Mario or Luigi is matched exactly with the optimal server, i.e., either $|o_M^i - a_M^i|$ or $|o_L^i - a_L^i|$ decreases by $d/2$. The term that does not decrease, however, may increase at most $d/2$, because that is the distance this friend walks. Hence, in the second case, the term $|o_M^i - a_M^i| + |o_L^i - a_L^i|$ remains exactly the same. In combination with the above observation that $|a_M^i - a_L^i|$ decreases by d, this concludes the proof.

Using the potential function, we can now prove the competitive ratio. Specifically, assume that for serving the ith request, OPT and ALG move Mario and Luigi a distance of d_O^i and d_A^i, respectively. Also, let $\Delta\Phi_i$ be the change of the potential function after request i, i.e., $\Delta\Phi_i = \Phi_i - \Phi_{i-1}$. Because in our case, we have $\Delta\Phi_i = 2d_0^i - d_A^i$ as shown above (one move by OPT and one move by ALG), it holds that

$$d_A^i + \Delta\Phi_i \;\leq\; 2 \cdot d_0^i.$$

Summing this term up over all iterations i, we obtain

$$\sum_i d_A^i + \sum_i (\Phi_i - \Phi_{i-1}) \;\leq\; 2 \cdot \sum_i d_O^i.$$

In the term $\sum_i (\Phi_i - \Phi_{i-1})$, all but the first and last term cancel out and hence,

$$ALG + \Phi_n - \Phi_0 \;\leq\; 2 \cdot OPT.$$

Because Φ_n is positive and Φ_0 is a constant, it follows that $ALG \;\leq\; 2 \cdot OPT + O(1)$. □

Note that the method of using a potential function is very powerful when dealing with online algorithms. In particular, whenever (for every possible online problem) we can come up with a potential function for which we can prove that an algorithm ALG fulfils properties i), ii), and iii), then ALG has a constant approximation of C, where C is the constant used in property ii).