ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2010 Prof. R. Wattenhofer / J. Smula, T. Langner

Discrete Event Systems

Solution to Exercise Sheet 10

1 Competitive Analysis

Competitiveness

In the script there is a definition for the competitiveness of an algorithm. However, this definition only holds if we want to evaluate an algorithm by means of its costs. Sometimes, we want to compare algorithms regarding their *benefit* rather than their costs. In this case, we have to be a bit careful with the definition of a c-competitive algorithm. We say that an algorithm ALG is c-competitive, if for all finite input sequences the solution of algorithm ALG is at most a factor c worse than the optimal algorithm, regardless of the algorithms being compared concerning costs or benefit.
According to whether we evaluate an algorithm based on costs or benefit, an algorithm ALG is c-competitive if for all finite input sequences I

$$\text{cost}_{\text{ALG}}(I) \leq c \cdot \text{cost}_{\text{OPT}}(I) + k \quad \text{or}$$

$$\text{benefit}_{\text{ALG}}(I) \geq \frac{1}{c} \cdot \text{benefit}_{\text{OPT}}(I) - k \quad \text{respectively.}$$

Competitive Analysis

The *competitive analysis* of an algorithm ALG consists of two separate steps. First, we show that for an arbitrary problem instance, the result of ALG is asymptotically at most a factor c worse than the the optimal *offline* result. This yields an upper bound on ALG's result, that is $\text{cost}_{\text{ALG}} \leq c \cdot \text{cost}_{\text{OPT}} + k$. If the task is to show that ALG is c-competitive for a constant c, then we are done. If we are interested in a tight analysis, we have to show that there is a problem instance where the result of ALG is a factor c worse than the optimal *offline* result. This gives a matching lower bound on the objective value of the algorithm, $\text{cost}_{\text{ALG}} \geq c \cdot \text{cost}_{\text{OPT}}$.

Naturally, the second step is easier than the first one because we *just* have to find a "bad instance". The first step is often much more involved. A pattern that works quite often is the following.

1. Consider an arbitrary input sequence for ALG.
2. Partition the input sequence into *suitable* parts.
3. Show that $c_{\text{ALG}} \leq c \cdot c_{\text{OPT}}$ for each part.

The tricky part here is to find a *suitable* partition in step 2.

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 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.

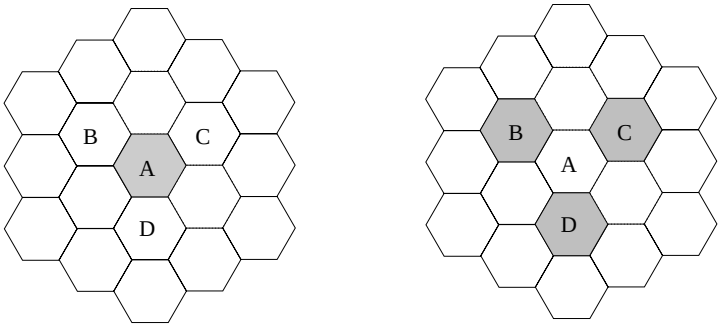

Figure 1: The solutions GREEDY (left) and OPT (right)

By accepting a call, 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. GREEDY accepts the first and has benefit 1. 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. We show this, again, by designing a "cruel" input sequence. 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 calls 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) = β, for an arbitrarily large value of β.

2 Power-Down Mechanisms

As mentioned in the hint, we only focus on a single idle period because if we know that our algorithm is c-competitive for any idle period, we also know that it is c-competitive for the complete busy sequence.

a) Analogously to the 2-competitive ski-rental online algorithm, we consider an algorithm ALG that powers down after D time units. To see that ALG is 2-competitive, we distinguish two cases for the length of the current idle period T:

- $T < D$: The energy consumed by both algorithms is $c_{\text{ALG}} = c_{\text{OPT}} = T$, hence the competitive ratio is $c = T/T = 1$.
- $T \geq D$: We have $c_{\text{ALG}} = D + D$ since ALG waits D time units and then powers-down and $c_{\text{OPT}} = D$ because OPT powers down immediately. Hence we get

$$c = \frac{2D}{D} = 2 \ .$$

b) Let ALG be any *deterministic* power down algorithm. Then ALG is fully specified by the time t_{ALG} after which it powers down in an idle period. The "worst" idle period ends immediately after ALG has powered down, that is we have $T = t_{\text{ALG}} + \varepsilon$. Again, we distinguish two cases with respect to the time t_{ALG} when ALG powers down.

- $t_{\text{ALG}} < D$: We have $c_{\text{ALG}} = t_{\text{ALG}} + \varepsilon + D$ and $c_{\text{OPT}} = t_{\text{ALG}} + \varepsilon$, hence

$$c = \frac{t_{\text{ALG}} + \varepsilon + D}{t_{\text{ALG}} + \varepsilon} = 1 + \frac{D}{t_{\text{ALG}} + \varepsilon} > 2 \quad \text{for } \varepsilon \to 0$$

  since $t_{\text{ALG}} < D$.

- $t_{\text{ALG}} \geq D$: We have $c_{\text{ALG}} = t_{\text{ALG}} + \varepsilon + D$ again and $c_{\text{OPT}} = D$, hence

$$c = \frac{t_{\text{ALG}} + \varepsilon + D}{D} = 1 + \frac{t_{\text{ALG}} + \varepsilon}{D} \geq 2 \quad \text{for } \varepsilon \to 0$$

  since $t_{\text{ALG}} \geq D$.

Hence, ALG cannot be better than 2-competitive.

c) Let ALG be a randomised algorithm that powers down at time $\frac{2}{3}D$ with probability $\frac{1}{2}$ and at time D otherwise. Let C_{ALG} be a random variable for the cost incurred by the algorithm. We again consider an arbitrary idle period of length T. We distinguish three cases:

- $T < \frac{2}{3}D$: The energy consumption of both algorithms is $c_{\text{ALG}} = c_{\text{OPT}} = T$, hence $c = T/T = 1 < 2$.

- $\frac{2}{3}D \leq T < D$: The expected energy consumption of ALG is

$$\mathbf{E}[C_{\text{ALG}}] = \frac{1}{2}\left(\frac{2}{3}D + D\right) + \frac{1}{2}T = \frac{5}{6}D + \frac{1}{2}T$$

  and further $c_{\text{OPT}} = T$. Hence we get

$$c = \frac{\frac{5}{6}D + \frac{1}{2}T}{T} = \frac{1}{2} + \frac{5}{6} \cdot \frac{D}{T} \leq \frac{1}{2} + \frac{5}{6} \cdot \frac{D}{\frac{2}{3}D} = \frac{1}{2} + \frac{5}{4} = \frac{7}{4} < 2 \ .$$

- $T \geq D$: We have for the expected energy consumption of ALG

$$\mathbf{E}[C_{\text{ALG}}] = \frac{1}{2}\left(\frac{2}{3}D + D\right) + \frac{1}{2}(D + D) = \frac{5}{6}D + D = \frac{11}{6}D$$

  and further $c_{\text{OPT}} = D$. Hence we get

$$c = \frac{\frac{11}{6}D}{D} = \frac{11}{6} < 2 \ .$$

Hence, the randomised algorithm is $\frac{11}{6}$-competitive which is better than any deterministic algorithm.

Note: This result, however, is not optimal yet. The best randomised algorithm uses a continuous probability distribution for the shutdown time and thereby achieves a competitive ratio of $e/(e-1) \approx 1.58$.