ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

Prof. Dr. R. Wattenhofer

Discrete Event Systems
Online Algorithm Exercise

Mario und Luigi sind Brüder und arbeiten als Eiscremeverkäufer an einem ein Kilometer langen Strandabschnitt an der Adria.[1] Immer wenn ein Badetourist ein Eis kaufen möchte (Request), muss mindestens einer der beiden zu dem Touristen eilen, und ihm ein Eis verkaufen. Da der Strand sehr flach und übersichtlich ist, können Mario und Luigi immer den ganzen Strand überblicken. Das heisst, bei einem neuen Request kennen Mario und Luigi immer sofort den Ort des Requests. Requests kommen nacheinander und die Zeit zwischen zwei aufeinanderfolgenden Requests ist immer genug lange, damit der Weg zum Kunden zurückgelegt kann, bevor der nächste Request gestellt wird. Wir nehmen also an, dass es nie zwei unerfüllte Requests gibt.

In Bild ist eine Beispielsituation dargestellt. Der $i$te Request befindet sich rechts von Luigi. Im Beispiel wird dieser Request von Luigi bedient. Dieser legt dafür einen Weg von 0.2 Kilometern zurück. Der $i+1$te Request liegt zwischen Luigi und Mario. Wer soll ihn bedienen?

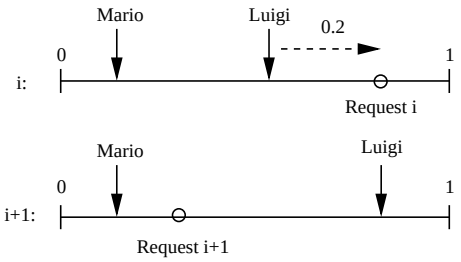

Figure 1: Der Strand mit Mario und Luigi's Position. Der $i$te Request wird von Luigi bedient.

Nehmen Sie an, dass Mario am Anfang auf Position 0 und Luigi auf Position 1 stationiert sind. Ein Request i ist definiert durch seine Position r_i im Interval $[0, 1]$. Mario und Luigi werden konfrontiert mit einer Sequenz von Requests $\sigma = r_1, r_2, \ldots$.

Sei d_M resp. d_L der gesamte Weg, den Mario resp. Luigi an einem Tag (es gibt potentiell unendlich viele Requests) zurücklegen. Um am Abend nicht müde zu sein, möchten Mario und Luigi den Weg, den sie zusammen zurücklegen so kurz halten wie möglich. Das heisst, sie möchten den Wert $d_M + d_L$ minimieren. Mit welcher Strategie sollten Mario und Luigi die Requests bedienen?

a) Mario und Luigi lassen sich von der renommierten Consulting-Firma Cons-ULT beraten. Diese schlagen vor, dass immer derjenige den Request i bedienen soll, der momentan näher bei r_i ist. Der andere soll einfach an seinem gegenwärtigen Ort bleiben und warten. Gemäss Cons-ULT ist dies die bestmögliche Strategie für Mario und Luigi.

   Verifizieren Sie die Aussage von Cons-ULT anhand folgender Request Sequenz σ_1:

$$\sigma_1 = 0.4\ ,\ 0.1\ ,\ 0.5\ ,\ 0.7\ ,\ 0.0$$

   Beschreiben Sie anhand von Skizzen den Weg den Mario und Luigi zurücklegen. Was wäre die optimale Lösung für Mario und Luigi gewesen? Wie kompetitiv ist der Cons-ULT-Algorithmus bezüglich der Sequenz σ_1?

[1] Unfortunately, Mario and Luigi cannot speak English fluently and we therefore deem it appropriate to pose this exercise in German.

b) Ist die Strategie von Cons-ULT c-kompetitiv für irgendeine Konstante c? Wenn ja, für welche Konstante? Beweisen Sie ihre Antwort.

c) Es gibt einen 2-kompetitiven Algorithmus ALG für das Problem von Mario und Luigi. Wie der Cons-ULT Algorithmus ist auch ALG deterministisch, also nicht randomisiert. Wie könnte dieser 2-kompetitive Algorithmus aussehen? Schlagen Sie einen deterministischen Algorithmus vor und begründen Sie, warum dieser 2-kompetitiv ist.

Sample Solution

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.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.

In general, it holds that Mario is always to the left of Luigi, both in our algorithm and in the optimal algorithm: This is true without loss of generality, because if the optimal solution switches the positions of the two friends, we can simply "rename" Luigi and Mario in order to reestablish the invariant.

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.

For iii), 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 $\Delta\Phi_i \leq 2d_O^i - d_A^i$ as shown above (one move by OPT and one move by ALG), it holds that

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

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 $\Phi_n \geq 0$ and $\Phi_0 = 1$, it follows that $ALG \;\leq\; 2 \cdot OPT + 1$. □

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).