Decentralized Mechanisms: Pricing Online Decisions

Allan Borodin, Denis Pankratov (Modified by Nic Bolton)

Here we will provide two related examples showing how costs can incentivize an online agent to simulate a centralized decision making algorithm resulting in substantially improved social welfare.

The first example is the deterministic DC algorithm (Algorithm 5.2.1 in subsection 5.2.3) for the k-server problem restricted to the line metric. This second example concerns a simplified parking problem which is modelled as a min cost bipartite matching problem.

The k-Server Problem for the Line Metric

We observe that the natural deterministic greedy algorithm for the k-server problem on the line would result in an arbitrarily bad competitive ratio.

Claim 9.3.1. *The natural greedy algorithm (i.e., serve a request by the nearest server) will result in an arbitrarily bad competitive ratio*

Proof. Suppose $y_1 < y_2 < \cdots < y_n$ are the metric space points and let $|y_3 - y_2| > |y_2 - y_1|$. We can use repeated requests to force $k-1$ servers to occupy locations $y_3, ..., y_n$. This is then followed by an arbitrarily long sequence of alternating requests to y_1 and y_2. This will result in one server alternating between y_1 and y_2 so that the optimum cost is constant (independent of the number of requests) while the greedy algorithm continues to pay $|y_1 - y_2|$ for each request. □

We recall from Section 5.2.3 that the lazy and non-lazy DC algorithms are k-competitive. We also know that any k-server algorithm can be made into an equally competitive lazy algorithm (where only one server moves to serve a request) by using "virtual locations" for the servers to know where the servers are located in the non-lazy algorithm. In particular, the lazy DC algorithm can be derived from any non-lazy algorithm achieving the same competitive ratio. Moreover, if the transformation to a lazy algorithm is done carefully (in terms of which server will be chosen when there is a tie) for the DC algorithm, then the lazy version will inherit two nice properties ("locality" and "monotonicity") from the non-lazy DC algorithm. Namely, a request will always be served by an adjacent server (locality), and if a new request r would be served by a server s, then every request in the closed interval between r and s would be served by s (monotonicity).

We now suppose that requests for service at a location r are being made by agents arriving online. The agent wants to minimize its cost for service which we can initially think of as the distance that some server will have to travel. Now we know that if agents act greedily and choose the closest server, the overall social welfare (that is, the total cost for all requests and hence the average time per server) can suffer arbitrarily. We want to show how simulate the lazy DC algorithm by dynamically assigning a cost $p(s)$ to each server s so that when an agent decides to use server s, the total cost to get service at location r will become $d(s, r) + p(s)$. Now when an online agent arrives and demands service at location r, the greedy decision is to minimize the sum of the

distance to the desired location plus the cost to the server. With an appropriate pricing mechanism, the resulting greedy online algorithm will "weakly simulate" the lazy DC algorithm and inherit the k-competitiveness of the DC algorithm. Here "weakly simulate" means that even though the lazy DC algorithm satisfies the locality and monotonicity properties stated above and has a well defined way to break ties, greedy agents are free to break ties unpredictably. However, it can be argued that no matter how a greedy agent breaks ties, the server movement costs of the simulating algorithm will be no more than the server movement costs of the lazy algorithm.

Theorem 9.3.2. *For the lazy DC algorithm (satisfying the locality and monotonicity properties), there exists a deterministic payment $p(s)$ for using a server s such that a greedy agent (incurring cost $d(s,r)+p(s)$ to serve a request r by the server s) will weakly simulate the lazy DC algorithm and the server movement costs will be at most that of the lazy DC algorithm.*

Proof. Consider the lazy DC algorithm and the mapping of servers to locations on the line after a request r has been served. We can assume that these are distinct location (after some initial sequence of k distinct requests). This mapping induces a partition of the line into intervals of possible requests that would be served by each server. Let $I(s)$ be the interval that would be served by s. If s and s' are adjacent severs, then $I(s)$ and $I(s')$ are adjacent intervals. Furthermore, for each pair of adjacent intervals s, s', there is a threshold point $\tau(s,s')$ such that any request in $[s,\tau(s,s'))$ will be served by s and requests in $[\tau(s,s'),s')$ will be served by s'. Let the left to right order of servers be $s_1,\ldots,s_k$. Once we set the price $p(s_1)$ for the left most server s_1, we can then set the remaining server prices so as to satisfy the following equation:

$$p(s_i)+|s_i-\tau(s_i,s_{i+1})| = p(s_{i+1})+|s_{i+1}-\tau(s_i,s_{i+1})| \tag{1}$$

□

Claim 9.3.3. *When agent acts greedily using the above payment, the location of the servers and the distance movement costs will weakly simulate the lazy DC algorithm. In particular, if agents only serve a request by an adjacent server and break ties in the same way as the lazy DC algorithm then the distance costs will be bounded by the distance costs of the lazy DC algorithm.*

Proof. Suppose we have a request $r \in I(s)$ where s is an arbitrary server. Consider another server $s' \neq s$ and the sequence of servers between s and s', namely $(s_0 = s', s_1, \ldots, s_n = s)$ such that the interval of each server $I(s_i)$ intersects the interval $[s',r]$. Using (1), for $0 \leq i < n$ we have

$$\begin{aligned} p(s_i) &= p(s_{i+1})+|s_{i+1}-\tau(i,i+1)|-|s_i-\tau(i,i+1)| \\ &\geq p(s_{i+1})-|s_{i+1}-s_i| \end{aligned}$$

This is done via triangle inequality (which is an equality in our case of a line metric), i.e., $|s_i-s_{i+1}| = |s_i-\tau(i,i+1)|+|\tau(i,i+1)-s_{i+1}|$. We then take a summation on both sides, where

for any $j \leq n$ we obtain

$$\begin{aligned}
\sum_{i=0}^{j-1} p(s_i) &\geq \sum_{i=0}^{j-1} \left(p(s_{i+1}) - |s_{i+1} - s_i|\right) \\
\sum_{i=0}^{j-1} p(s_i) &\geq \sum_{i=1}^{j} p(s_i) - \sum_{i=0}^{j-1} |s_{i+1} - s_i| \\
p(s_0) &\geq p(s_j) - |s_j - s_0| \\
\implies p(s_j) &\leq p(s') + |s' - s_j| \qquad (2)
\end{aligned}$$

We arrive at two cases. The first is if $s_n = s \in [s', r]$, then (2) can be used to derive $p(s') + |s' - r| = p(s') + |s' - s| + |s - r| \geq p(s) + |s - r|$, which shows how the agent making request r is incentivized to choose s as this choice minimizes the cost that the agent inherits.

The other case is where $s_n = s \notin [s', r]$. Note that $s \notin [s', r] \iff r \in [s', s)$. Since s_{n-1} and $s_n = s$ are adjacent, we have $|s - \tau(s_{n-1}, s)| \geq |s - r|$ which allows us to derive

$$\begin{aligned}
p(s) + |s - r| &\leq p(s) + |s - \tau(s_{n-1}, s)| \\
&= p(s_{n-1}) + |s_{n-1} - \tau(s_{n-1}, s)|
\end{aligned}$$

Recall that $r \in I(s) \implies |s_{n-1} - \tau(s_{n-1}, s)| \leq |s_{n-1} - r|$:

$$\begin{aligned}
p(s) + |s - r| &\leq p(s_{n-1}) + |s_{n-1} - \tau(s_{n-1}, s)| \\
&\leq p(s_{n-1}) + |s_{n-1} - r|
\end{aligned}$$

Finally, apply (2) to get $p(s_{n-1}) + |s_{n-1} - r| \leq p(s') + |s' - s_{n-1}| + |s_{n-1} - r| = p(s') + |s' - r|$. Thus we arrive at the same result, namely $p(s) + |s - r| \leq p(s') + |s' - r|$, where (just as in the first case) the agent is incentivized to choose s.

We have shown that for any agent making a request r, it always is incentivized to choose the server whose interval contains r (i.e., s such that $r \in I(s)$). By assuming every agent will break ties the same way that lazy DC does, then we conclude the server locations and the distance movement costs will weakly simulate lazy DC. □

Claim 9.3.4. *Even if the greedy agents do not break ties as does the lazy DC algorithm, there is a modified payments scheme that will guarantee the same asymptotic bound on the movement costs and hence the competitive ratio will be k.*

Proof. Consider a request r located somewhere between two adjacent servers s_i, s_{i+1}. We are interested in the scenario where a choice of either server results in minimizing the cost inherited by the agent making the request, i.e., when $|s_i - r| + p(s_i) = |s_{i+1} - r| + p(s_{i+1})$.

This scenario occurs when the request r is made at $\tau(s_i, s_{i+1})$ and we get back the original pricing scheme as defined in (1). In this case, whether the agent chooses s_i or s_{i+1} is nonetheless acceptable (in the context of simulating lazy DC), as either could be matched with r in a minimum cost matching.

All that is left for consideration is the other non-adjacent servers. The consideration applies to all pairs of successive servers s_j, s_{j+1} whose threshold point $\tau(s_j, s_{j+1})$ does not lie strictly between them (i.e., it is located exactly on either of s_j or s_{j+1}). If $\tau(s_j, s_{j+1}) = s_j$ and r arrives

at location s_j, the agent is indifferent between using server s_j (a local move) and s_{j+1} (a non-local move). If the agent breaks the tie by choosing s_{j+1}, the locality property is violated, and the agent is no longer simulating lazy DC.

$$\begin{aligned} p(s_j) + |s_j - \tau(s_j, s_{j+1})| &= p(s_{j+1}) + |s_{j+1} - \tau(s_j, s_{j+1})| \\ p(s_j) &= p(s_{j+1}) + |s_{j+1} - s_j| \quad (\because \tau(s_j, s_{j+1}) = s_j) \\ &= p(s_{j+1}) + |s_{j+1} - r| \end{aligned}$$

We employ a perturbation argument by modifying the pricing scheme to ensure that for any pair of adjacent servers, the threshold vertex is strictly between them. Let $\epsilon > 0$ be an arbitrarily small constant. Whenever the pricing equation implies that $\tau(s_j, s_{j+1})$ coincides with a server location (e.g., s_j), we shift the threshold $\tau(s_j, s_{j+1})$ by ϵ towards s_{j+1}. Consequently, if a request is at location s_j then the cost to use s_j is now strictly less than the cost to use s_{j+1}. The greedy agent is forced to behave "locally", restoring the simulation of lazy DC.

Let the new threshold be $\tau'(s_j, s_{j+1}) = \tau(s_j, s_{j+1}) + \epsilon = s_j + \epsilon$. Then we must also have new prices to accomodate this

$$\begin{aligned} p'(s_j) + |s_j - \tau'(s_j, s_{j+1})| &= p'(s_{j+1}) + |s_{j+1} - \tau'(s_j, s_{j+1})| \\ p'(s_j) + \epsilon &= p'(s_{j+1}) + |s_{j+1} - s_j - \epsilon| \\ p'(s_j) + 2\epsilon &= p'(s_{j+1}) + |s_{j+1} - s_j| \quad (\because |s_{j+1} - s_j| > \epsilon) \end{aligned}$$

The agent will make their decision with these updated costs, specifically $p'(s_j) + |s_j - r| = p'(s_j)$ for s_j, and $p'(s_{j+1}) + |s_{j+1} - r| = p'(s_{j+1}) + |s_{j+1} - s_j|$ for s_{j+1}. Obviously the agent is incentivized to choose s_j now (just as lazy DC would), as:

$$p'(s_{j+1}) + |s_{j+1} - s_j| = p'(s_j) + 2\epsilon > p'(s_j)$$

We must account for the cost overhead introduced by this perturbation. For a sequence of n requests, we can choose a perturbation $\epsilon_k = \delta/2^k$ for the k^{th} request, where δ is a small constant. The total additional additive cost incurred by the mechanism is bounded by:

$$\sum_{k=1}^{n} \epsilon_k = \sum_{k=1}^{n} \frac{\delta}{2^k} < \sum_{k=1}^{\infty} \frac{\delta}{2^k} = \delta$$

Since the total cost is strictly less than arbitrarily small δ and is independent of the number of requests n, this additive constant does not affect the asymptotic competitive ratio. Thus, even with adversarial tie-breaking by agents, we can enforce the simulation of lazy DC with negligible cost overhead. □

The Parking Problem on the Line

An obvious example where agents are creating online events is when drivers look for a legal place to park. They naturally want to park as near to their intended destination as possible. We will study a very simplified version of this problem. Namely, we will consider a static problem rather than drivers coming and going for different amounts of time. The static version is not totally unrealistic

as it does model the situation when say individuals are driving to an event and everyone leaves after that event or situations where parking is available for the evening (e.g., 7PM to 6AM). Many jurisdictions will charge for parking spaces. Note that if the charge is fixed for all spaces, then from our perspective, this is the same as free parking as drivers would only care about the distance between where they park and their desired location. Our final and main assumption is that the parking spaces are located on discrete points on the line and that there are enough parking spaces for all drivers.

More specifically, suppose we are given the locations of m offline points $\{x_i\}$ on the real line and let $d(x, y)$ denote the distance between x and y; that is, $d(x, y) = |x - y|$. Suppose $n \leq m$ agents (e.g., shoppers) arrive and will decide to park at some unoccupied location x_i so as to shop at some location y_i. A central authority (i.e., a mechanism) will determine a cost $p_i(x_i)$ for each parking spot x_i based on the current location of available spaces. An agent parking at x_i incurs a total cost of $d(x_i, y_i) + p_i(x_i)$. The social welfare objective is to minimize $\sum_{i=1}^{n} d(x_i, y_i)$. The social objective then becomes the special case of min cost metric matching restricted to the line metric. Of course, this is a different social objective than when the goal is to generate revenue for the central authority.

For the worst case competitive analysis of an algorithm, we can assume that $m = n$. An offline optimum algorithm can compute a min cost perfect matching in polynomial time for any weighted graph problem. For the special case of matching on the line, there is a very simple linear time optimal algorithm. Namely if the locations are sorted so that $x_1 \leq x_2 \ldots \leq x_n$ and $y_1 \leq y_2 \ldots \leq y_n$, then match x_i to y_i.

When an agent arrives, the natural greedy decision for an agent acting in self-interest is to choose an available (i.e., not yet occupied) location that is closest to their goal destination (without loss of generality, we can assume that all distances between adjacent vertices are distinct so as to not need a tie-breaking rule). This suggests two immediate questions:

1. What is the competitive ratio of the best online algorithm?

2. What is the competitive ratio when decisions are being made greedily by agents?

We will first observe that the natural greedy algorithm has a very poor performance.

Claim 9.3.5. *The competitive ratio for the natural greedy algorithm (and hence if agents act greedily) is exponential in n where n is the number of online requests.*

Proof. ~~Consider the following instance for any $\epsilon > 0$: $y_1 = 1 - \epsilon$, $y_i = 2^{i-1}$, $x_1 = 1.5$, $x_i = 2^{i-1}$ for $2 \leq i \leq n$. The optimum matching has cost $.5 + \epsilon$ (for matching x_1 with y_1 and 0 for matching $x_i = y_i$ for $i \neq 1$). However, the greedy solution will match x_1 with y_2 forcing x_i to match with y_{i+1} for $i = 2, \ldots, n-1$ and x_n to match with y_1. The cost of the greedy solution will be $\approx 2^{n+1}$.~~ [1]

Consider the instance where we have $y_i = 2^i + 1$ for $1 \leq i \leq n$, and $x_1 = 0$, $x_i = 2^i + 1$ for $2 \leq i \leq n$. Suppose the agents arrive in order such that they wish to visit y_1, then y_2, and so on. The optimal solution is to match x_i with y_i (for all i), which results in a total cost of 2 (for matching x_1 with y_1, and 0 for the remaining).

1. I don't believe this approach is correct. The greedy solution would not match x_1 with y_2 – it would produce a matching that is the same as the optimal solution, i.e., x_i with y_i for all i.

Now consider the greedy solution. The first arriving agent (who wishes to visit y_1) will park at x_2, since $|y_1 - x_2| = 2 < |y_1 - x_1| = 3$. For the agents wishing to visit $y_2, \cdots, y_{n-1}$, the agent visiting y_i will look to park at x_i. Since x_i is occupied by the $(i-1)^{th}$ agent, it will park at the next nearest parking spot. The nearest two adjacent and available parking spots (relative to y_i) are at x_1 and x_{i+1}. The agent will choose x_{i+1}, since $|y_i - x_{i+1}| = 2^i < |y_i - x_1| = 2^i + 1$. The only agent who has not been considered at this point is the one visiting y_n, who's only choice is to park at x_1. The total cost of the greedy solution is thus

$$\begin{aligned}\sum_{i=1}^{n-1} |y_i - x_{i+1}| + |y_n - x_1| &= \sum_{i=1}^{n-1} \left|2^i + 1 - (2^{i+1} + 1)\right| + |2^n + 1 - (0)| \\ &= \sum_{i=1}^{n-1} 2^i + 2^n + 1 \\ &= 2^{n+1} - 1\end{aligned}$$

□

We want to again show that agents can be incentivized (by parking costs) to greedily park so as to achieve an $O(\log n)$ competitive ratio. The high level approach will be as follows:

- There is a natural (non-greedy) randomized online algorithm (Harmonic) that achieves a competitive ratio of $O(\log n)$. Harmonic is motivated by the deterministic double coverage algorithm for the k-server problem with respect to the line metric space and its canonical randomization (analogous to the canonical randomization in Algorithm 11.5.1 of the double sided greedy USM algorithm)
- There is a randomized dynamic pricing algorithm for parking locations that will incentivize the shopper to simulate the Harmonic algorithm and hence achieve a competitive ratio of $O(\log n)$. The pricing algorithm does not know the arrival location nor the desired location for the i^{th} arrival but only knows which locations are now occupied given the previous $i-1$ arrivals. When the i^{th} arriving agent chooses location x_i, their total cost will be $|x_i - y_i| + p_i(x_i)$