Introduction to Online Algorithms

Naveen Sivadasan

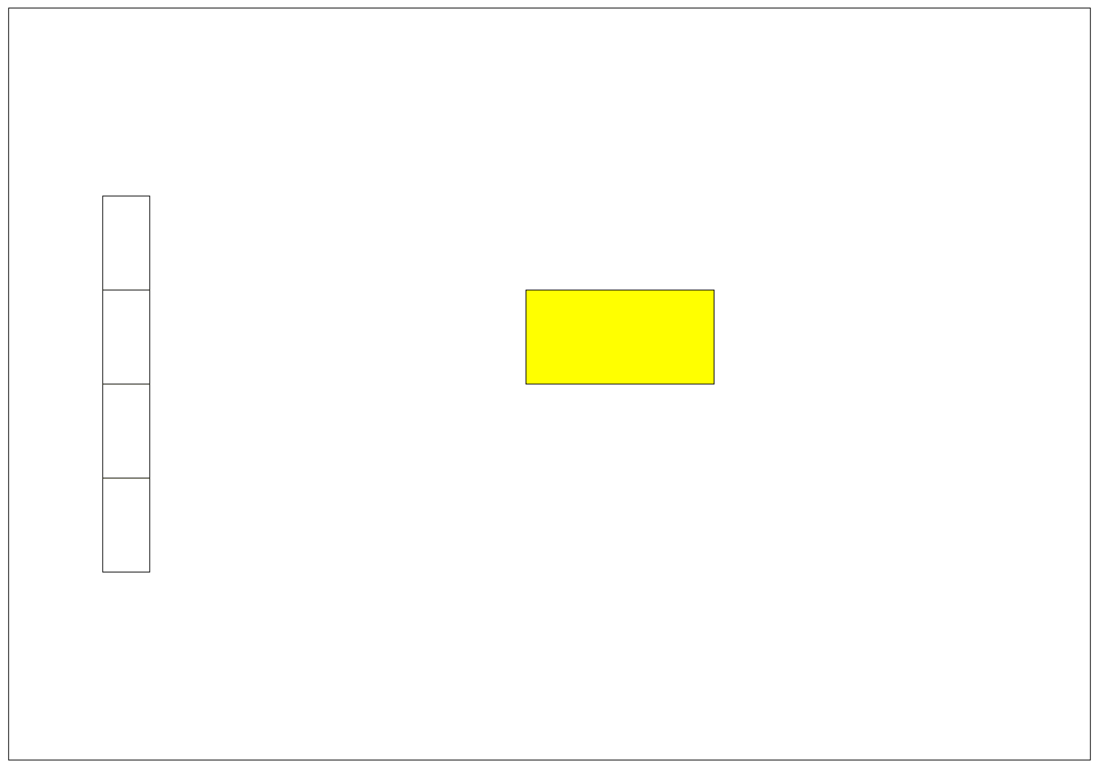
Online Computation

- In an online setting, the complete input is not known in advance.
- Input is a request sequence that is revealed gradually over time.
- Can be viewed as a request-answer game between the algorithm and an adversary.
- Algorithm has to perform well under the lack of information on future requests.
- Applications in scheduling, data structures, OS, networking etc.

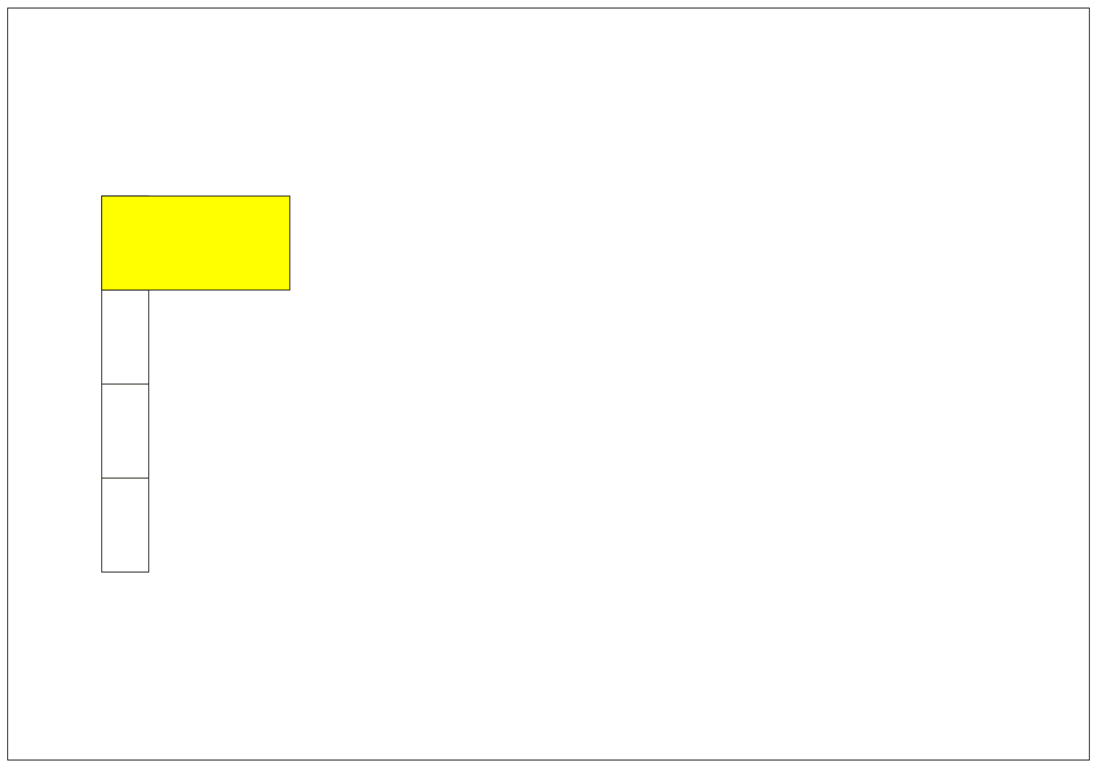
Online Makespan Scheduling

- Given m identical machines. That is, processing time for a job is same across all machines.
- Consider a sequence of requests $\sigma = j_1, j_2, j_3, \cdots, j_n$ of length n.
- Let j_i denote the processing time of job i.
- Each job j_i has to be assigned to exactly one machine. Once a job is assigned to a machine, it remains there.
- Objective is to minimize the completion time of the last finishing job (makespan).

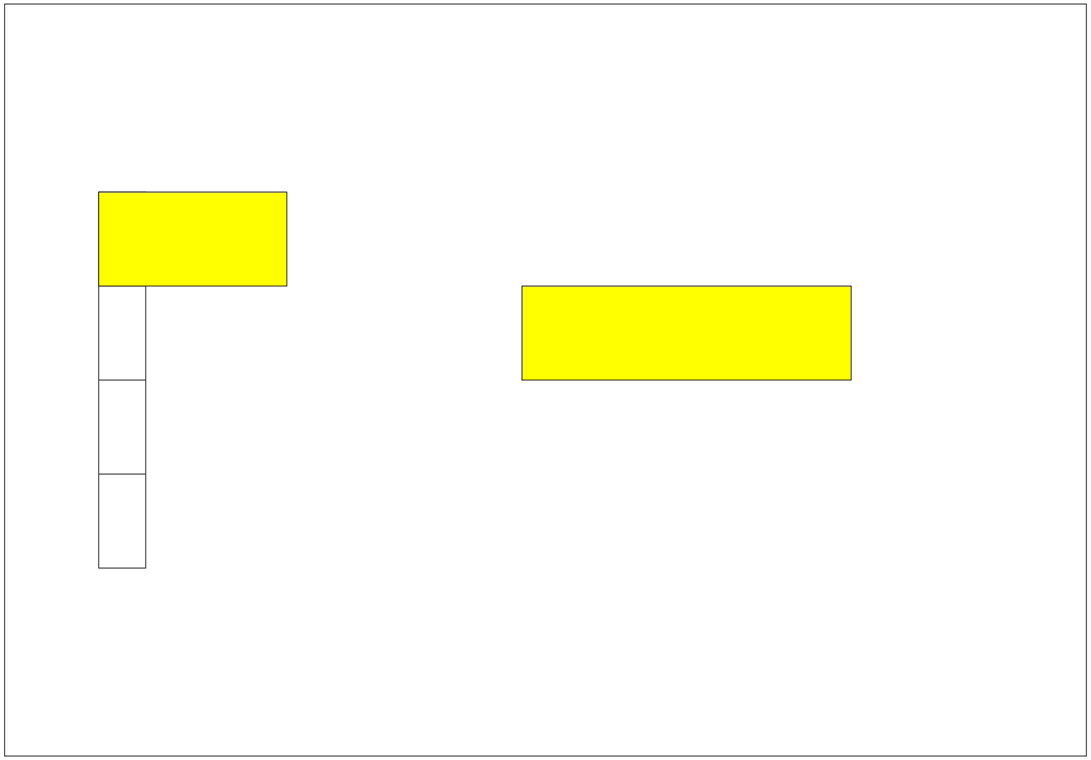
Example



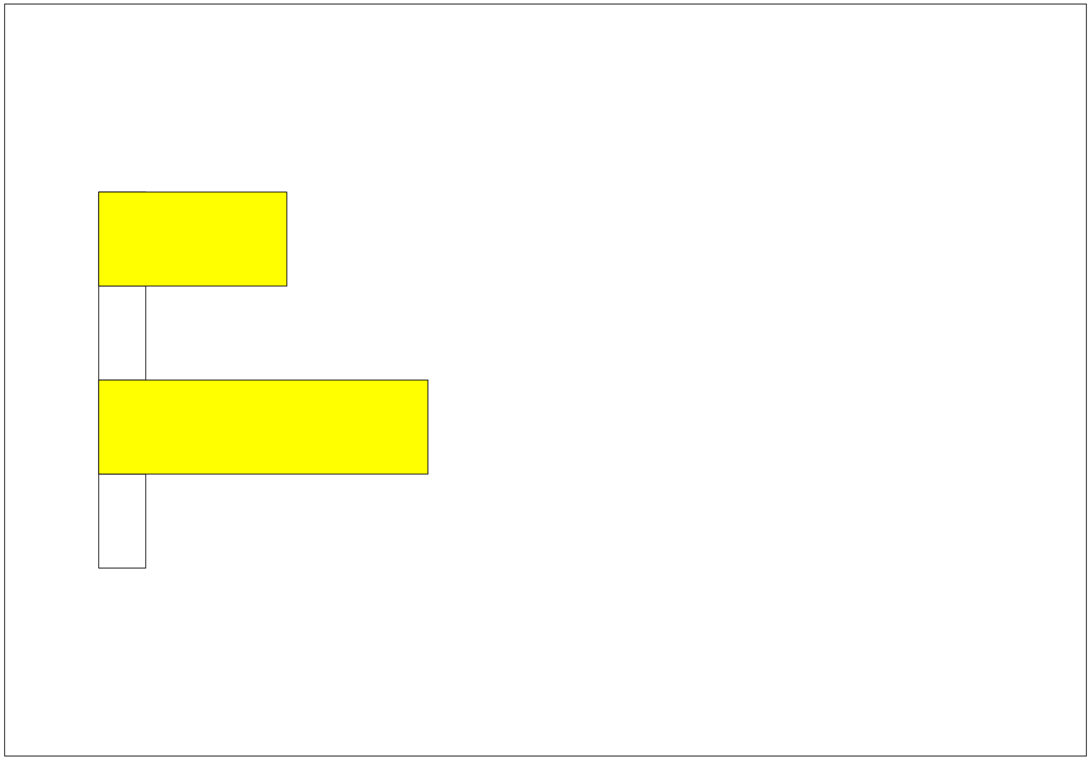
Example



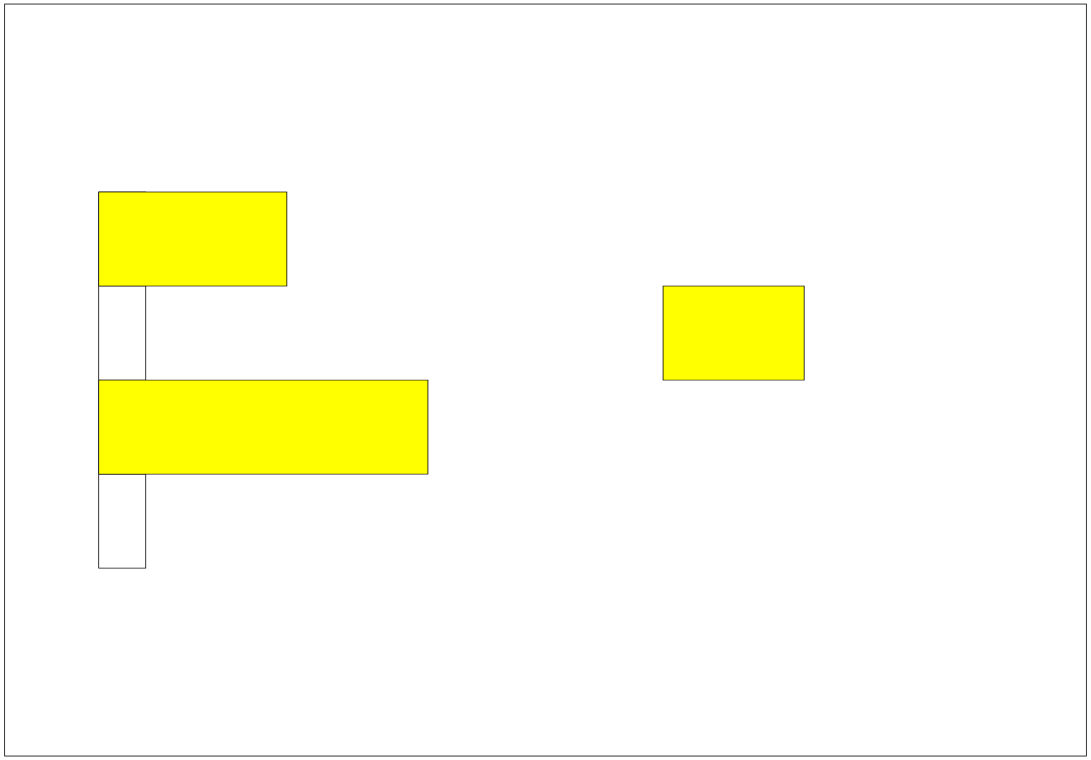
Example



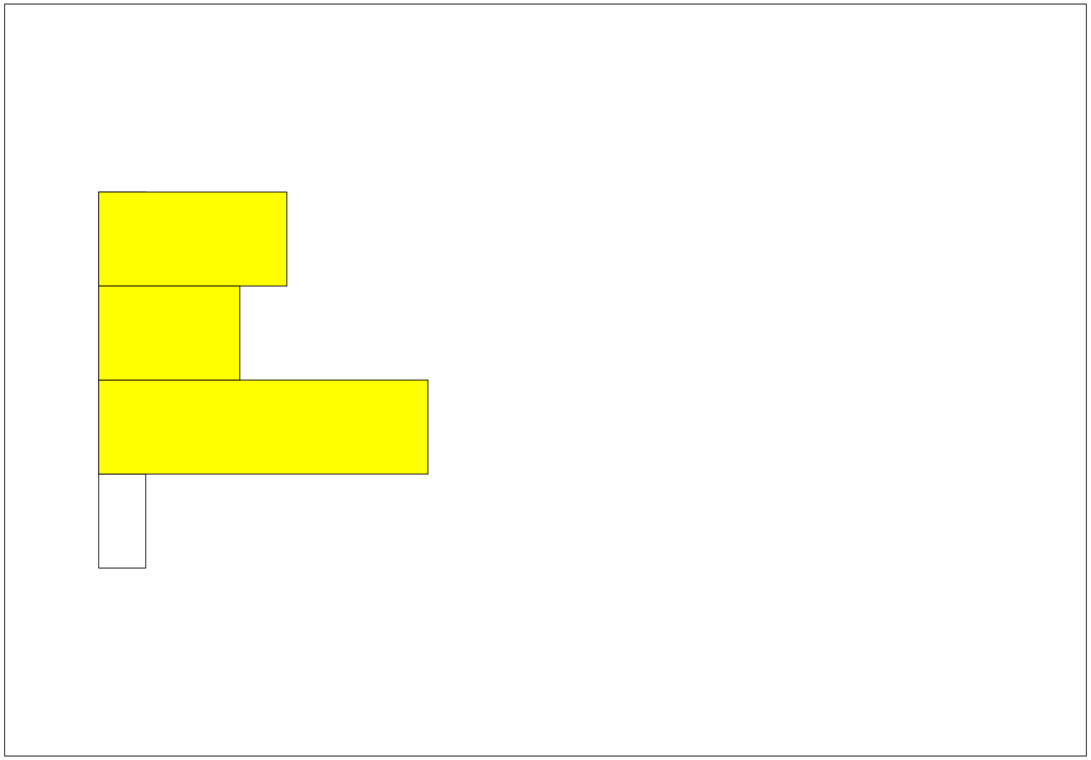
Example



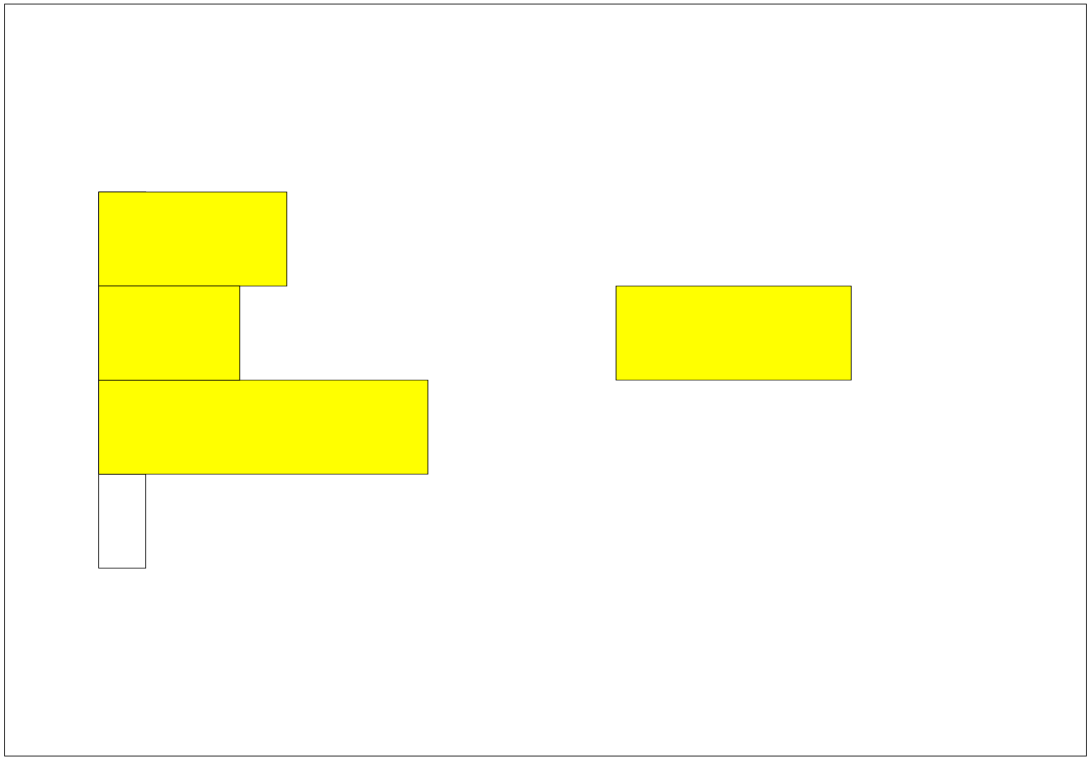
Example



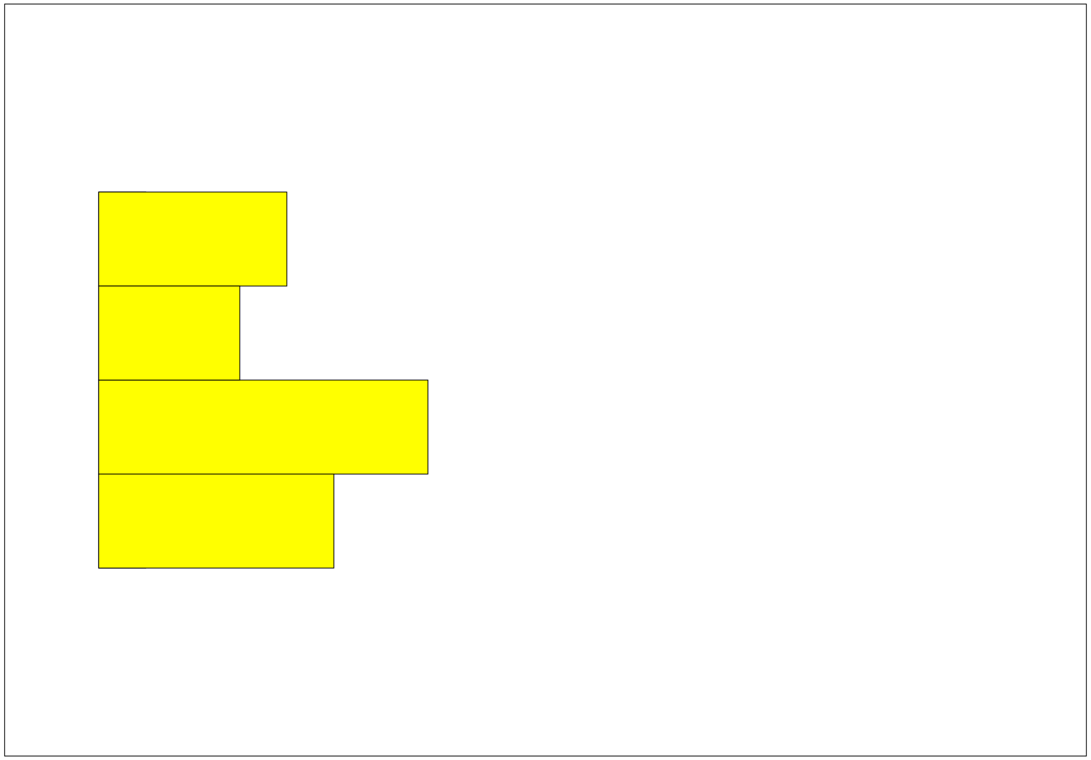
Example



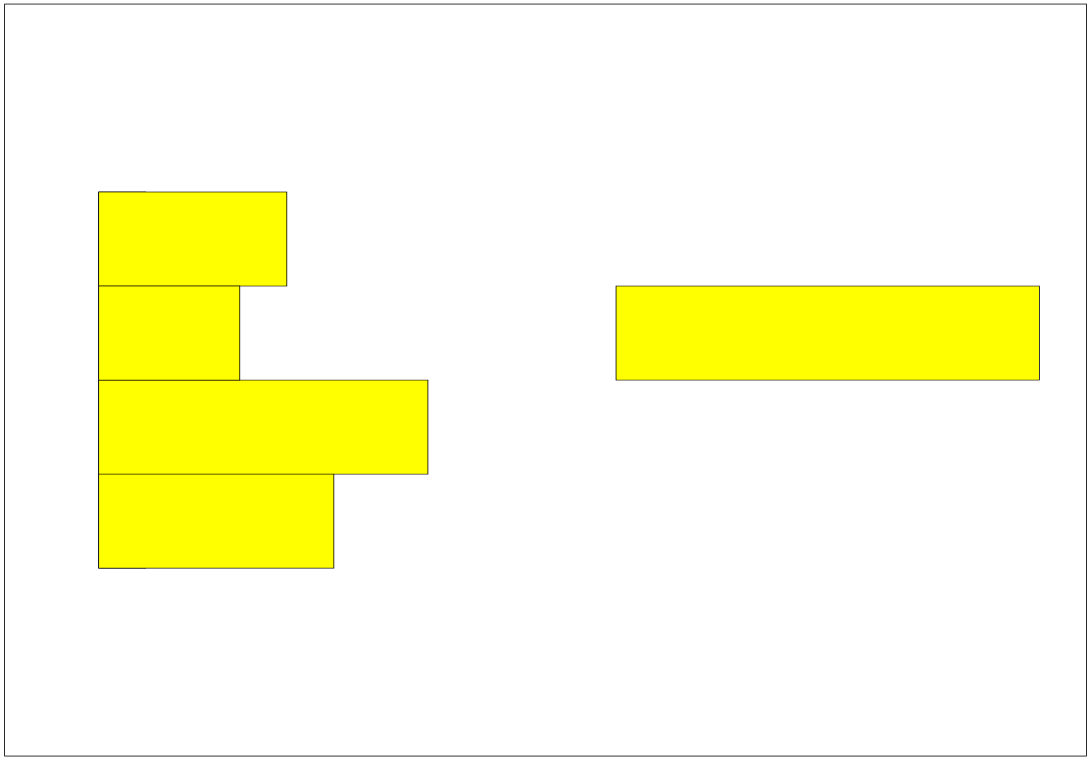
Example



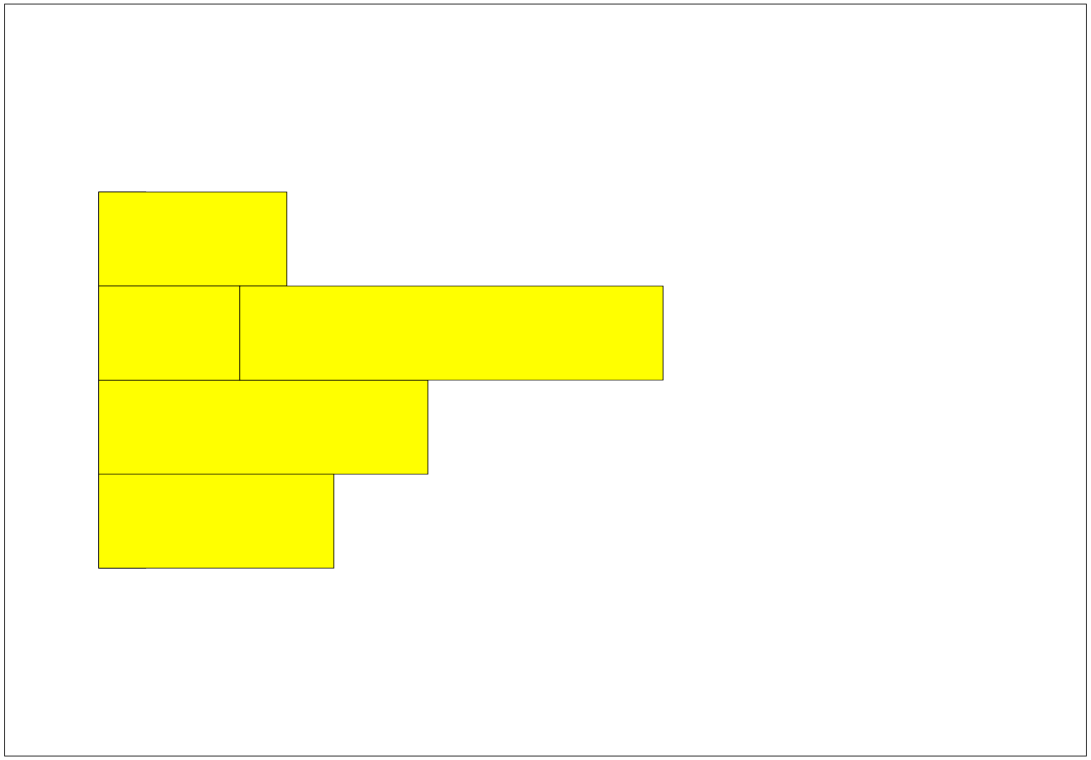
Example



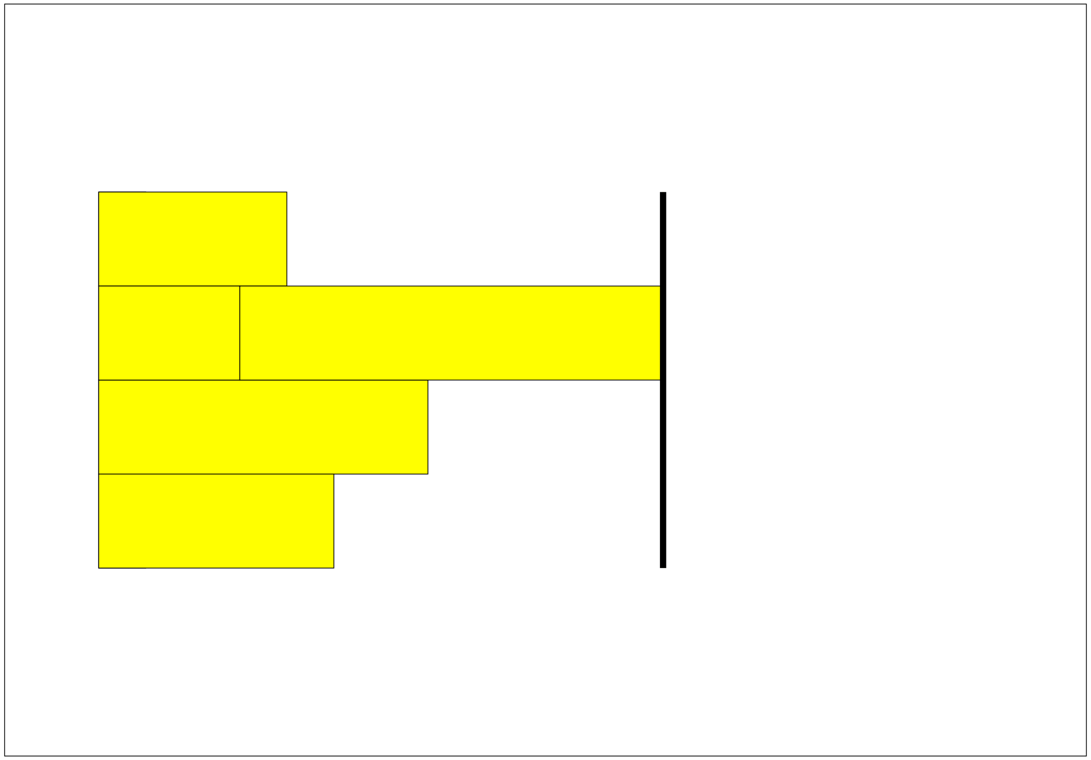
Example



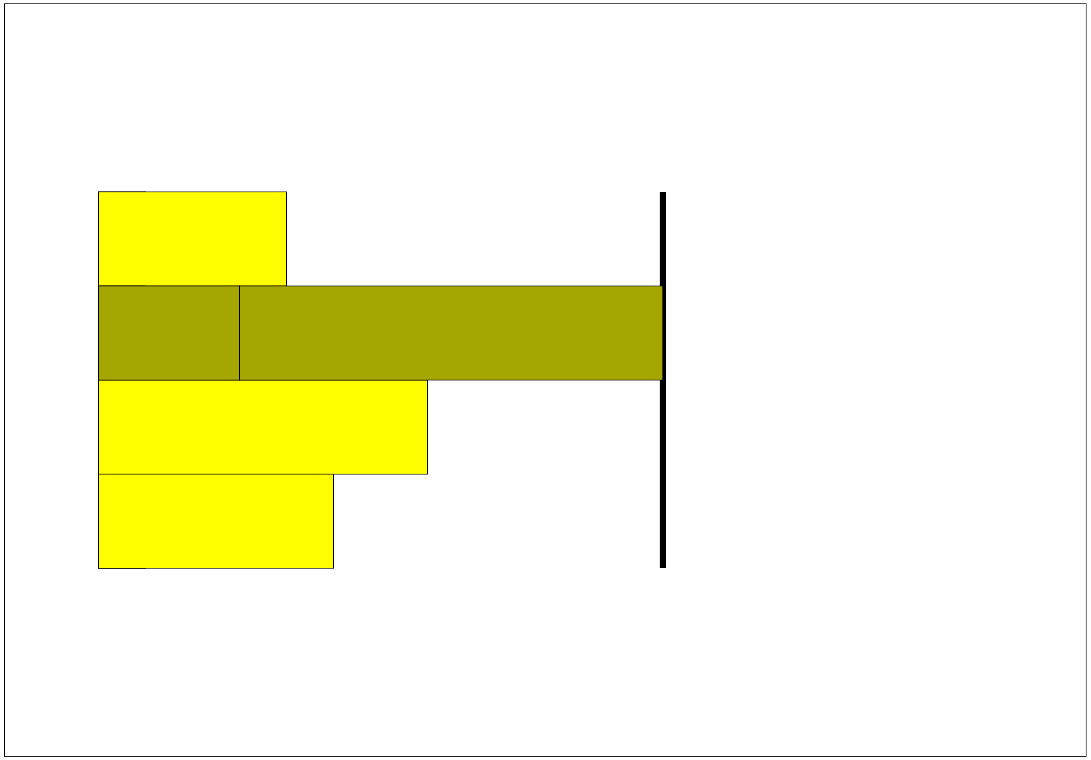
Example



Example



Example



Better Schedule

Competitive Ratio

- Compare the performance of the algorithm against offline optimal strategy.
- Let $\sigma = \sigma(1), \sigma(2), \sigma(3), \ldots, \sigma(t)$ denote a t length sequence.
- Request $\sigma(i)$ is revealed to the algorithm in round i.
- Let $A(\sigma)$ denote the cost incurred by the algorithm A for serving σ (Make span in prev. example)
- Let $OPT(\sigma)$ denote the optimal cost incurred if the complete σ is known in advance.
- A is said to be c–competitive if $A(\sigma) \leq c \cdot OPT(\sigma) + a$ for any sequence σ. (Here a is some fixed constant)

Back to Makespan Scheduling

Consider the following greedy approach :

- Schedule the new job to the least loaded machine. (Graham's list scheduling)
- The scheduling given in the previous example follows this approach.
- How competitive is this approach ?

Competitive ratio of greedy

- Consider any request sequence $\sigma = j_1, j_2, \ldots, j_n$.
- Focus on the makespan machine. Let w be the last job in it and r be the completion time excluding w. Hence $A(\sigma) = r + w$.
- When w was assigned greedily, all other machines also had load at least r.
- Hence $m \cdot r + w \leq j_1 + j_2 + \ldots + j_n$
- Observe that $OPT(\sigma)$ is at least the average load and also the size of any one job.
- That is, $\frac{m \cdot r + w}{m} \leq OPT(\sigma)$ and also $w \leq OPT(\sigma)$.
- Putting together, $A(\sigma) = r + w \leq 2 \cdot OPT(\sigma)$. (2-competitive)

Self-organizing lists

- Consider a list L of n elements $\{a_1, a_2, \ldots, a_n\}$.
- Cost of $access(x)$ (accessing an element x) in L is $rank(x)$.
- Algorithm is allowed to reorganize the list using paid exchanges with adjacent elements or move item to head of the list free of cost.
- Cost of an exchange is 1.
- Input is an online sequence $\sigma = x_1, x_2, x_3, \cdots$ of elements in L.
- Objective is to minimize the total cost of serving σ.

Self-organizing lists : Move To Front (MTF) algorithm

- Under standard worst case analysis, any algorithm would incur a cost of $|\sigma| \cdot n$ if each request is to access the last element of the list.
- This would not allow us to compare algorithms.
- Let analyze a simple, practical algorithm under online setting and do its competitive analysis.
- MTF (Move to Front): When an element is accessed, move it to front of the list.
- MTF incur a cost $rank(x)$ to access element x.
- Some accesses are cheap and some are costly. We require an aggregate cost analysis to account for this.

Amortized analysis

Binary counter example.

- An aggregate cost analysis technique.
- Consider a $\log n$ bit number that increment by 1 in one step.
- Cost of one increment is say the number of bit flips.
- What is the total cost (no. of bit flips) when number goes from $0 \cdots n-1$? (total $n-1$ steps)
- Clearly total cost is at most $n \log n$ bit flips. Better bound?
- Some increments are costly but many increments are cheap. Require an aggregate analysis to account for this.

Amortized analysis and Potential functions

Binary counter example. Bounding total number of bit flips.

- Number goes from $0 \cdots n-1$. (Total $n-1$ steps).
- A potential function Φ as a credit/debit mechanism to balance costly increments with savings from cheap increments.
- Define a potential function Φ_i that maps the state of the number after i increments to a non negative real number.
- Let Φ_i be the total number of ones in the number. Clearly $\Phi_0 = 0$.
- Amortized cost (bit flips) of ith increment defined as $\hat{C}_i = C_i + \Phi_i - \Phi_{i-1}$ where C_i is the actual cost (The potential difference takes care of credit/debit)
- Total amortized cost = $\sum_{i=1}^{n-1} \hat{C}_i$ and total actual cost = $\sum_{i=1}^{n-1} C_i$.
- We are only overestimating the total cost by amortized cost as $\sum_{i=1}^{n-1} \hat{C}_i = \sum_{i=1}^{n-1}(C_i + \Phi_i - \Phi_{i-1}) = \Phi_{n-1} + \sum_{i=1}^{n-1} C_i$

Amortized analysis and Potential functions

Binary counter example. Bounding total number of bit flips.

- We will now show that total amortized cost is $O(n)$, which is an upper bound on total actual cost.
- Consider one step from $XXXXX0111\cdots11$ to $XXXXX1000\cdots00$.
- Say there are k 1s in the end
- Actual cost is $k+1$; potential difference $\Phi_i - \Phi_{i-1}$ is $1-k$; amortized cost per step is hence 2.
- Total amortized cost is thus $O(n)$ for total $n-1$ steps.

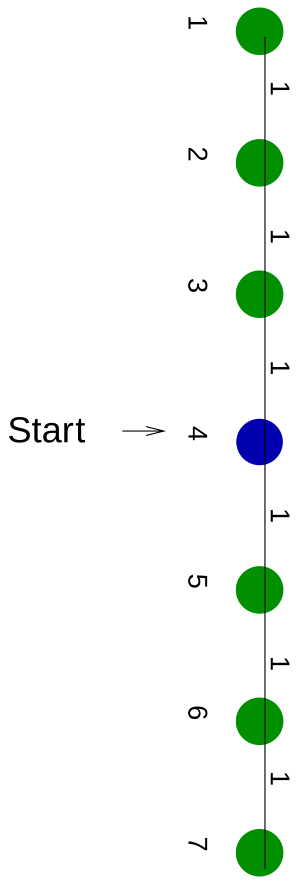
Amortized analysis - Move To Front (MTF) algorithm

- Let $C_{MTF}(t)$ denote the cost of MTF to serve tth request.
- Define potential Φ_t based on the difference between the MTF list and OPT list after round t.
- Define Φ_t as the number of inversions in MTF list with respect to OPT list.
- Number of inversions is the number of pairs that appear in opposite order in MTF list compared to OPT list.
- We will show that amortized cost $C_{MTF}(t) + \Phi_t - \Phi_{t-1} \leq 2C_{OPT}(t)$.
- This imply that MTF cost for whole input sequence is at most twice the OPT cost. That is MTF is 2-competitive.

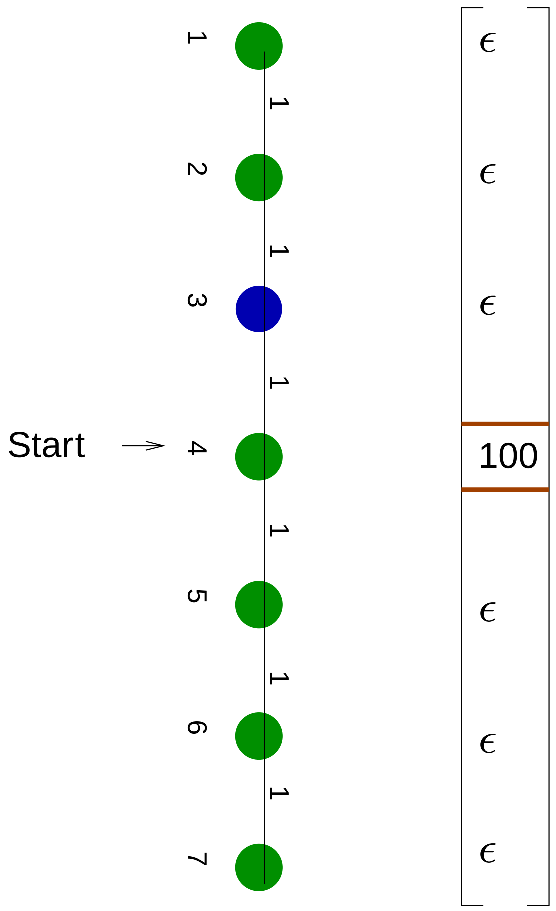
Amortized analysis - Move To Front (MTF) algorithm

- Let element accessed in round t be x.
- Let k denote the number of elements that precede x in both MTF list and OPT list.
- Let r denote the number of elements that precede x only in MTF list.
- We have $C_{MTF}(t) = k + r + 1$ and $C_{OPT}(x) \geq k + 1$.
- Moving x to front introduces k new inversions and destroys (saves) r old inversions. Thus $\Phi_t - \Phi_{t-1} = k - r$.
- Hence $C_{MTF}(t) + \Phi_t - \Phi_{t-1} = k + r + 1 + k - r = 2k + 1 \leq 2C_{OPT}(t)$.
- Each possible paid exchange by OPT increases potential by 1 but also increase cost of OPT by 1. Hence no issue.

The Metrical Task Systems Framework

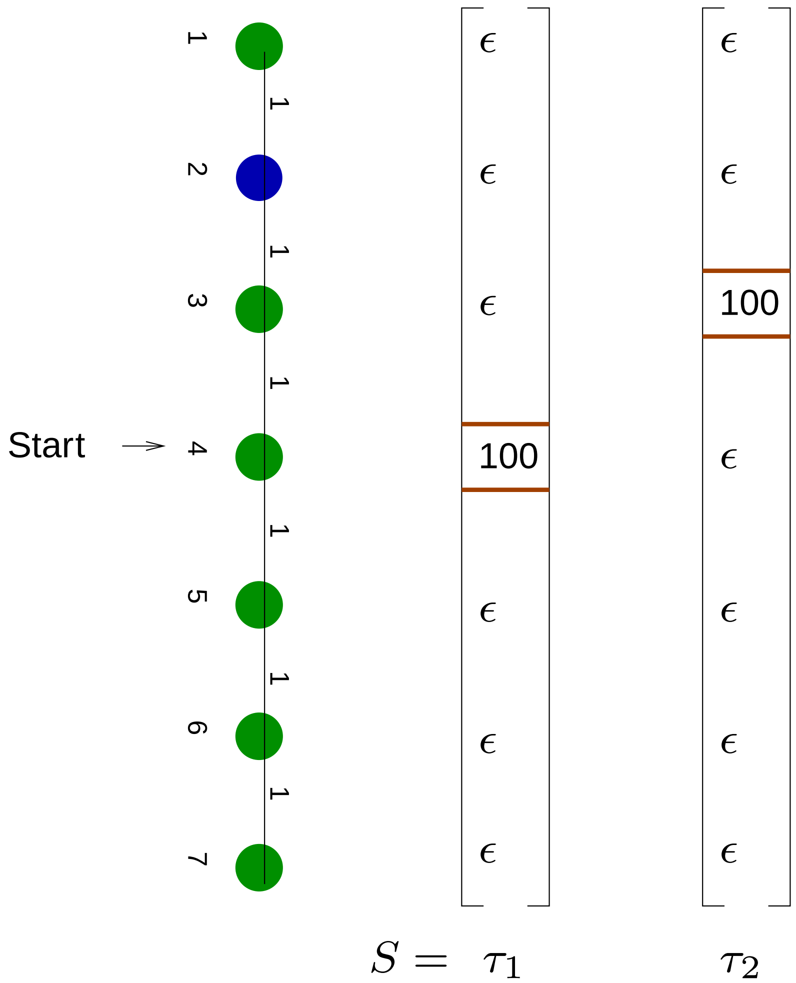


The Metrical Task Systems Framework

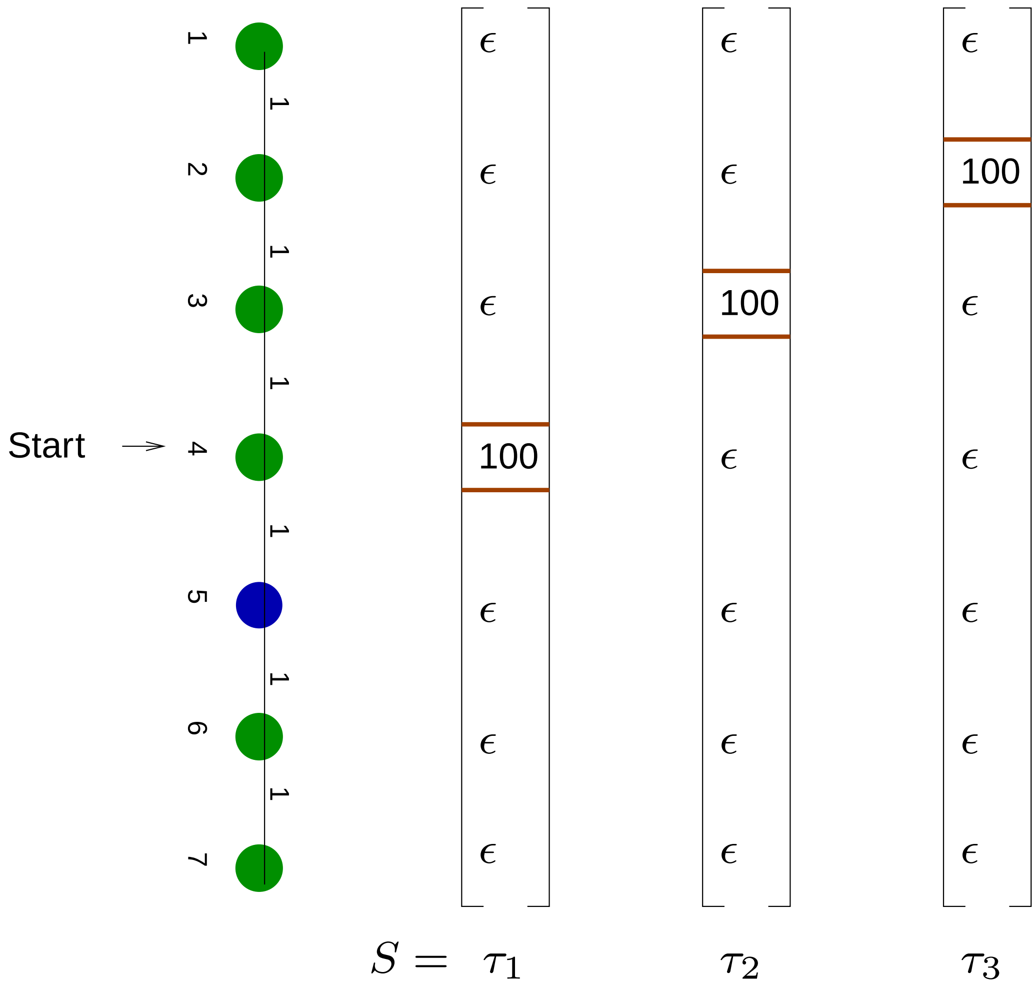


$S = \tau_1$

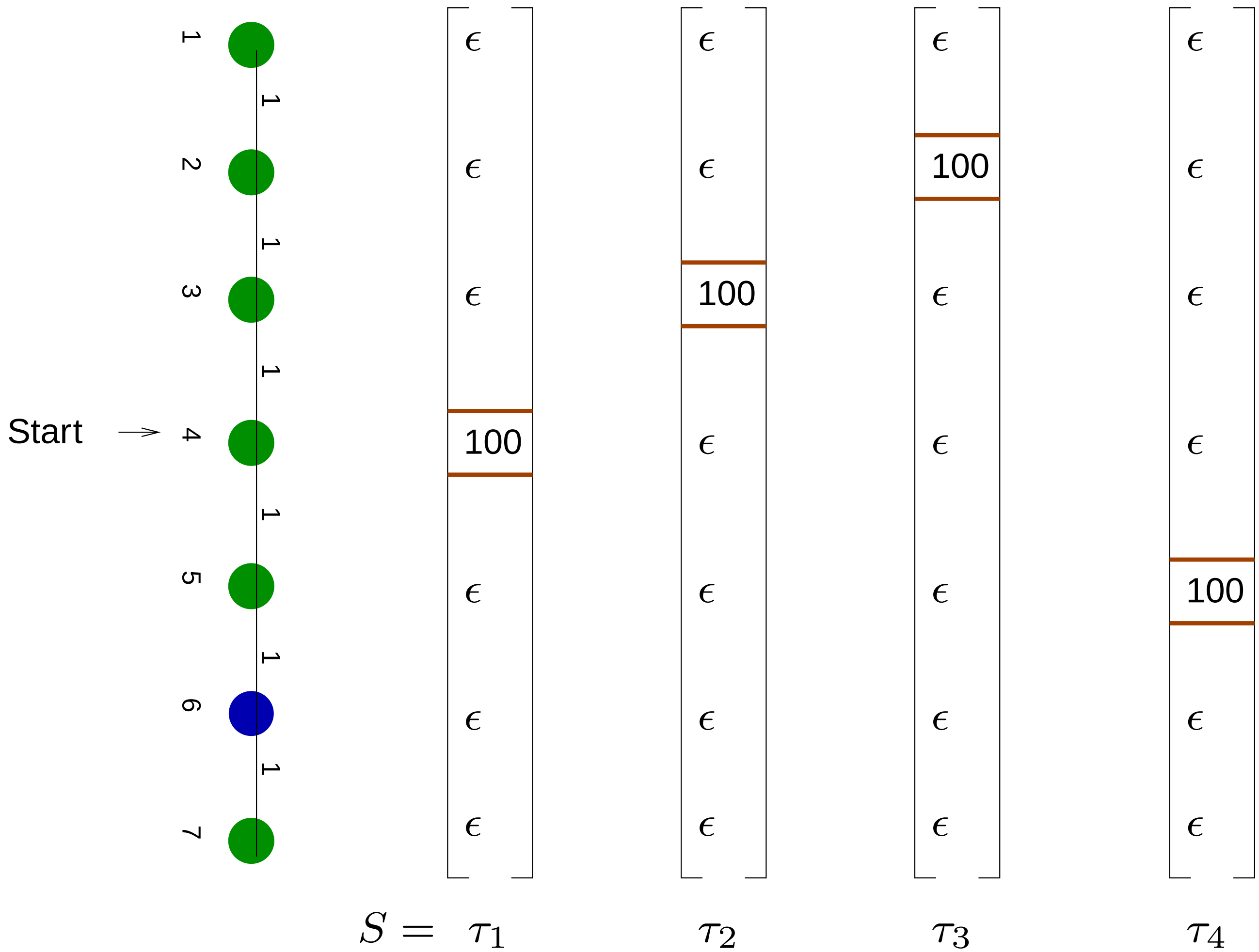
The Metrical Task Systems Framework



The Metrical Task Systems Framework



The Metrical Task Systems Framework



$$ALG[S] = 6 + 4\epsilon, \quad \text{opt}[S] = 2 + 4\epsilon. \quad \frac{ALG[S]}{OPT[S]} = \frac{6+4\epsilon}{2+4\epsilon} \approx 3$$

Metrical Task Systems (MTS) – Lower Bound

Theorem 1 *On any n state metric space and for any deterministic algorithm the* $\mathrm{c.ratio}$ *is at least* $2n - 1$*.*

That is, $\exists$ a bad adversarial instance for the specified graph and the specified algorithm.

There is an algorithm called work function algorithm that matches this lower bound.

Metrical Task Systems (MTS) – Lower Bound

- Fix any deterministic algorithm A.
- Consider $2n-1$ algorithms $\mathcal{B} = \{B_1, B_2, \ldots, B_{2n-1}\}$ such that the following invariant is always maintained.
- One alg from $\mathcal{B}$ occupy the same node as A and the rest of the nodes are occupied by exactly 2 algs from $\mathcal{B}$.
- If A makes a transition to vertex v from u in a round i, then one of the two algs from v moves to u. Thus invariant is maintained.
- Let v_t denote the node where A resides after t rounds.
- The adversarial input σ is such that in round t, processing cost at node v_{t-1} is ϵ and 0 everywhere else.

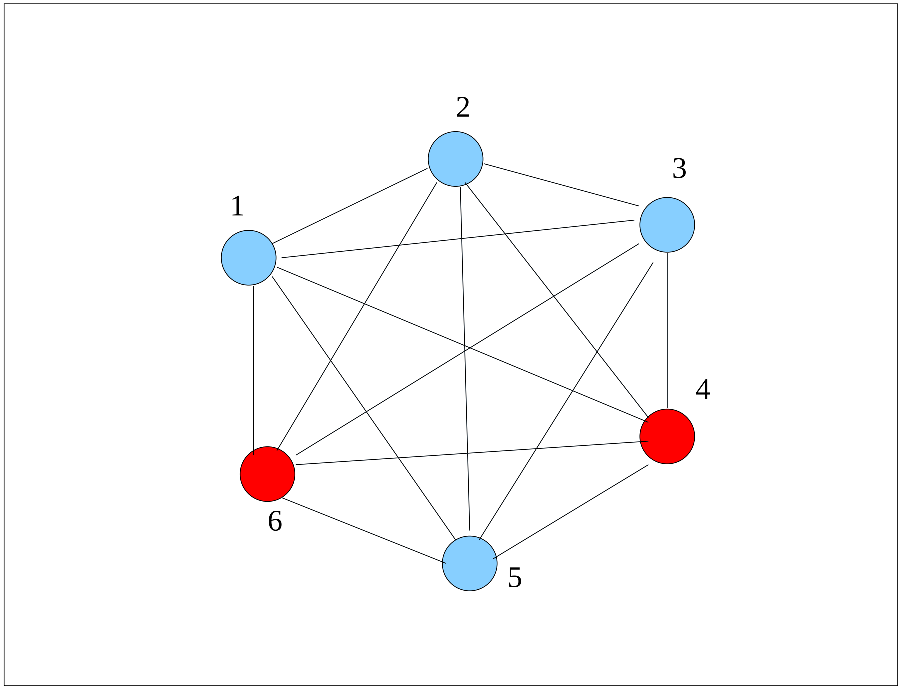
Metrical Task Systems (MTS) – Lower Bound

- Let $s = |\sigma|$.
- If A makes total k transitions to serve σ then $A(\sigma) = (s-k)\epsilon + T$, where T is the total travel cost.
- Let $\mathcal{B}(\sigma)$ denote the sum total of cost of all algs in $\mathcal{B}$, which is $\sum_{i=1}^{2n-1} B_i(\sigma)$
- Note that $\mathcal{B}(\sigma) = (s-k)\epsilon + T + 2k\epsilon = A(\sigma) + 2k\epsilon$.
- Also, $OPT(\sigma) \leq \frac{1}{2n-1}\mathcal{B}(\sigma)$.
- Hence $OPT(\sigma) \leq \frac{1}{2n-1}(A(\sigma) + 2k\epsilon) \leq \frac{1}{2n-1}A(\sigma)(1+2\epsilon)$.
- That is, $ALG(\sigma)/OPT(\sigma) \geq 2n-1$.

k-server problem

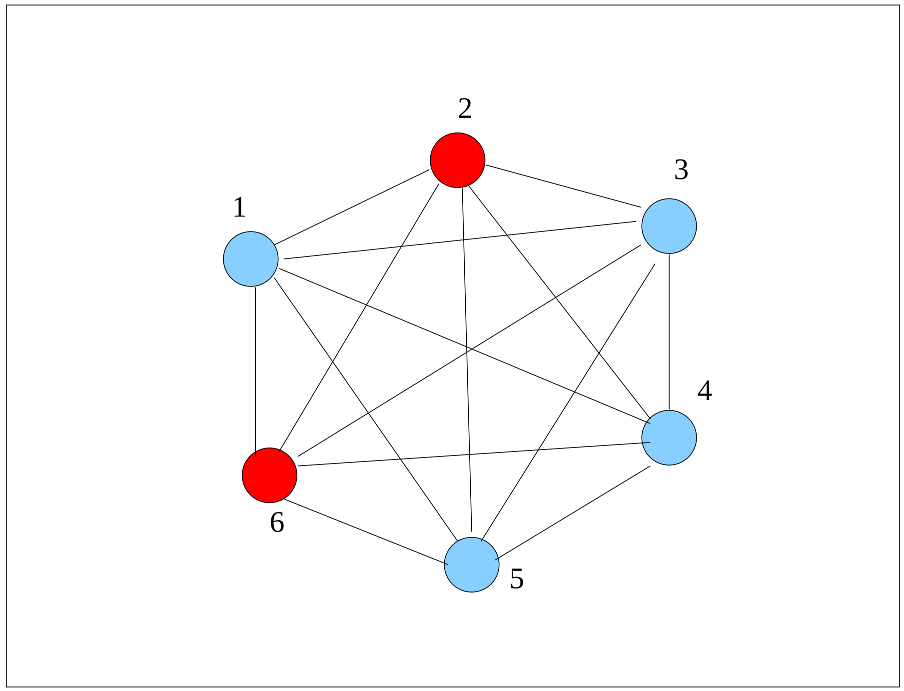
- There are k machines/servers that can move around in an n node weighted graph (which is a metric)
- No two servers reside in the same node.
- Each request $\sigma(i)$ is a node in the graph where the request should be served.

2–server



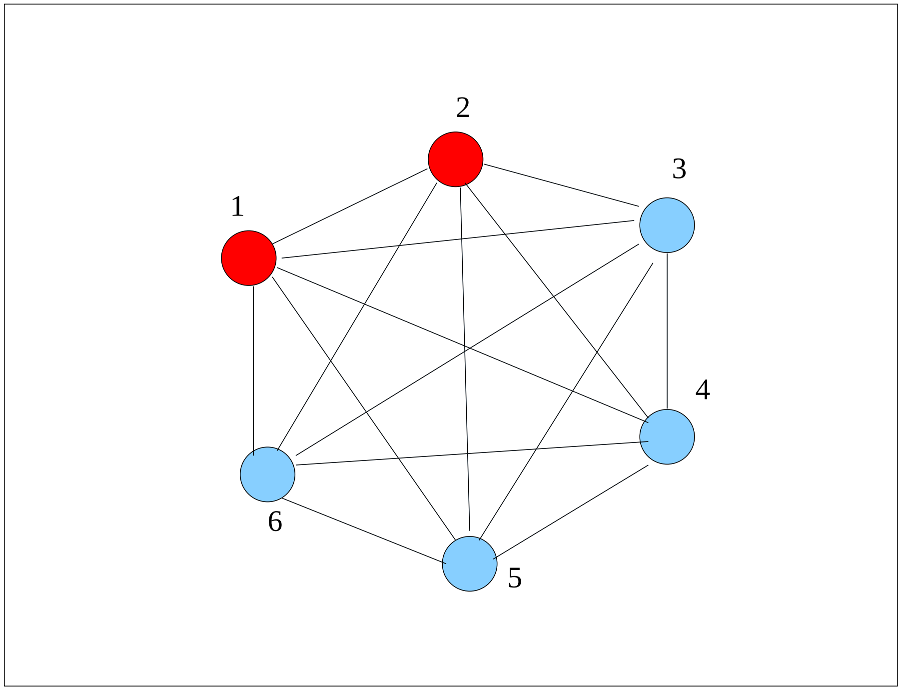
$\sigma =$

2–server



$\sigma = 2$

2–server



$\sigma = 2, 1$. $A(\sigma) =$ total travel cost for serving σ.

k-server problem

- There are k machines/servers that can move around in an n node weighted graph (which is a metric)
- No two servers reside in the same node.
- Each request $\sigma(i)$ is a node in the graph where the request should be served.
- Algorithm can send any of the k servers to serve the request.
- Cost incurred in a step is the distance the chosen server has to travel to serve the request.
- Total cost on a request sequence is the sum of the travel cost in each round.
- Generalization of problems such as paging problem.

k-server problem

- Actively researched area to bound the competitive ratio on arbitrary metric and on special cases.
- It is known that the best possible competitive ratio lies between k and $2k - 1$ for any arbitrary metric.
- It is still open whether the competitive ratio of the problem is exactly k.
- It is conjectured so.

References

[1] Allan Borodin and Ran El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge Univ. Press, Cambridge, 2005.

[2] S. Albers, *Online algorithms: A survey*, Mathematical Programming, 97:3-26, 2003.

[3] J. Sgall, *On-line scheduling - A survey*, Online Algorithms: The State of the Art, LNCS. 1442, pages 196-231, Springer, 1998.

[4] Elias Koutsoupias, *The k-server problem*, Survey, 2009.