Introduction to Online Algorithms

Naveen Sivadasan

Indian Institute of Technology Hyderabad

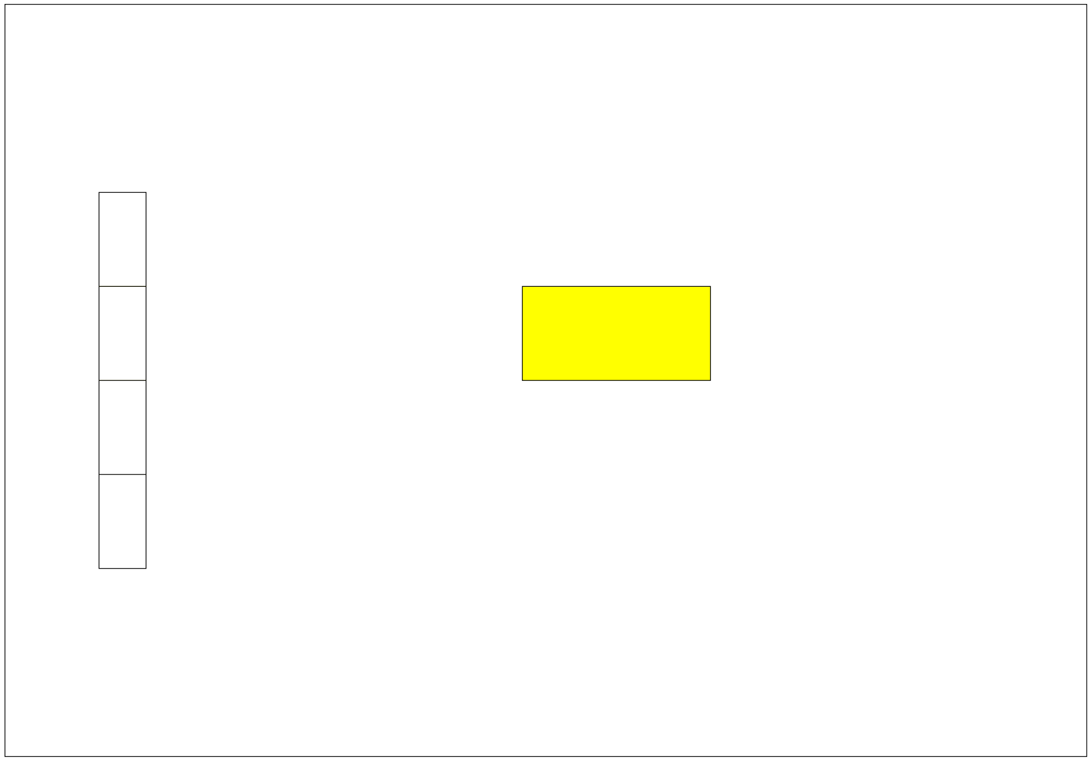
Online Computation

- 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.

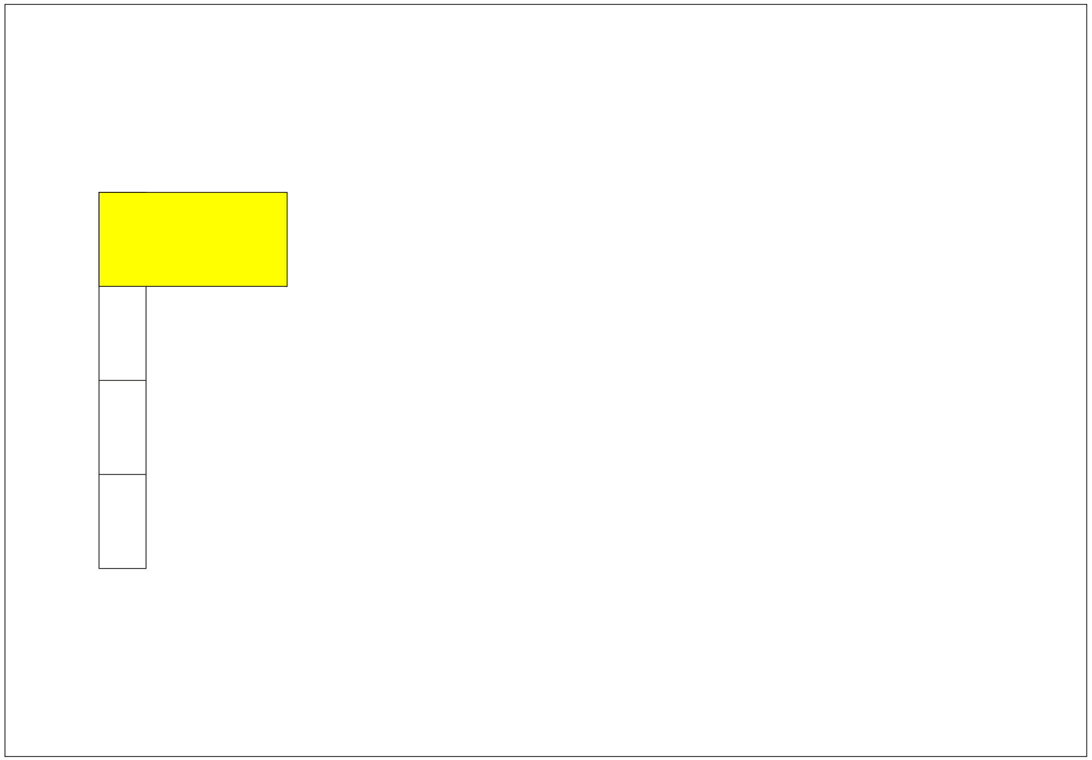
Online Makespan Scheduling

- 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).

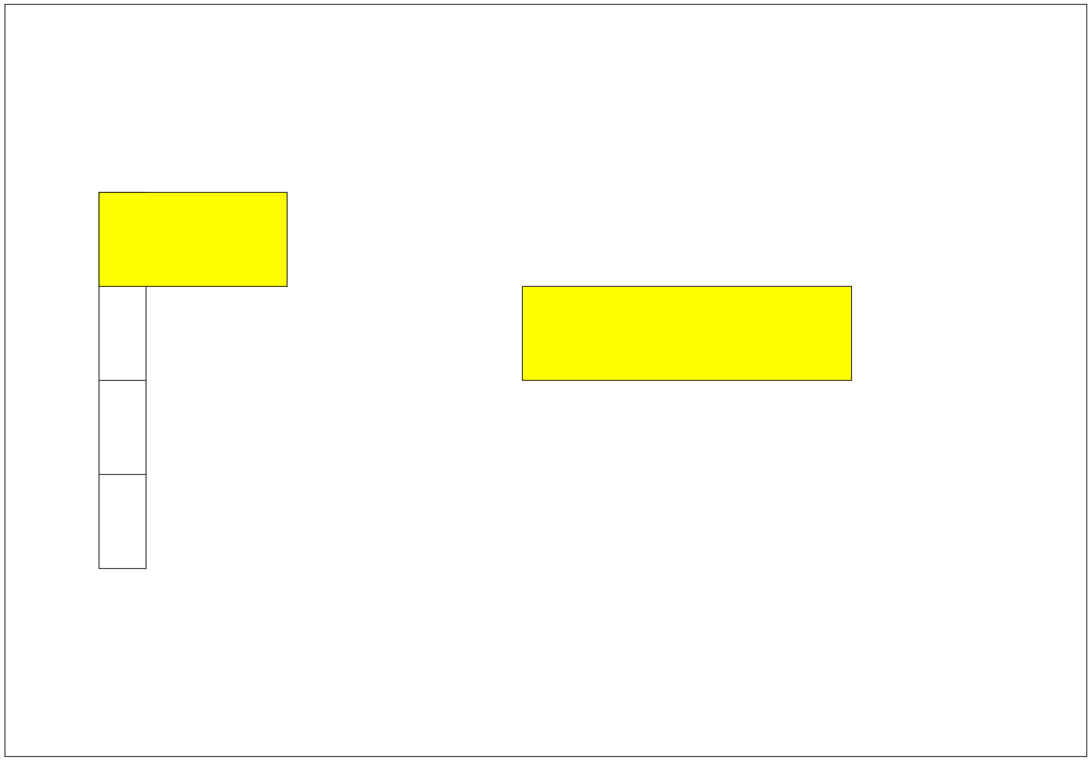
Example



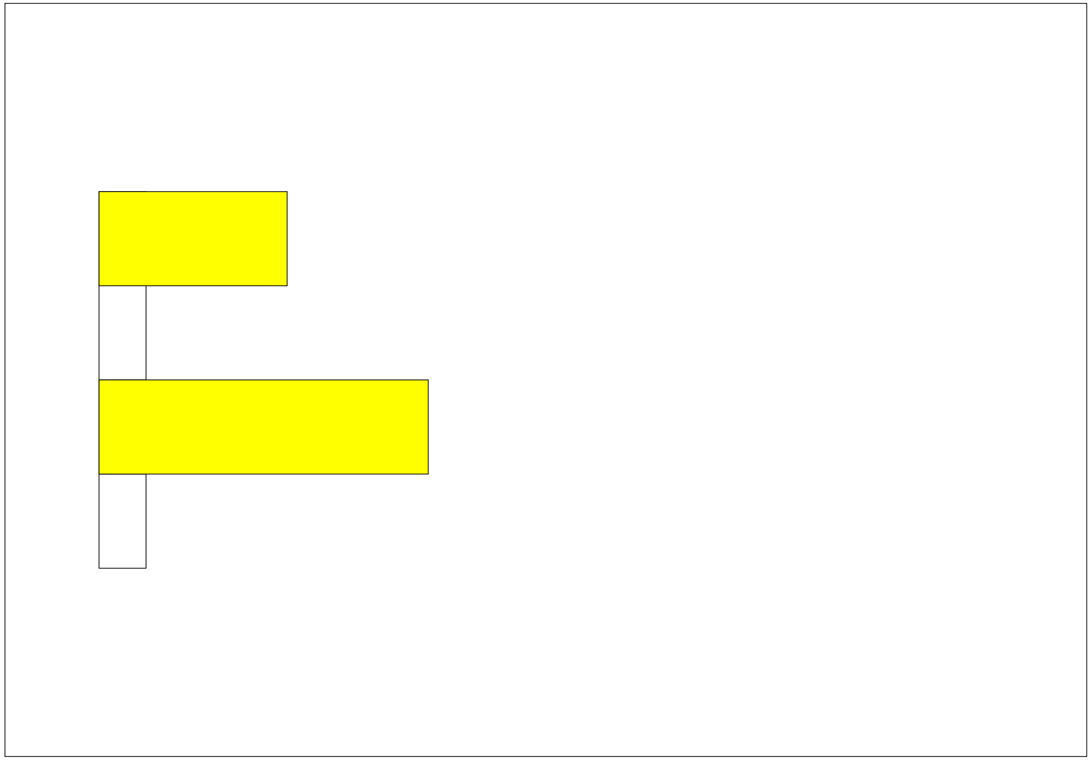
Example



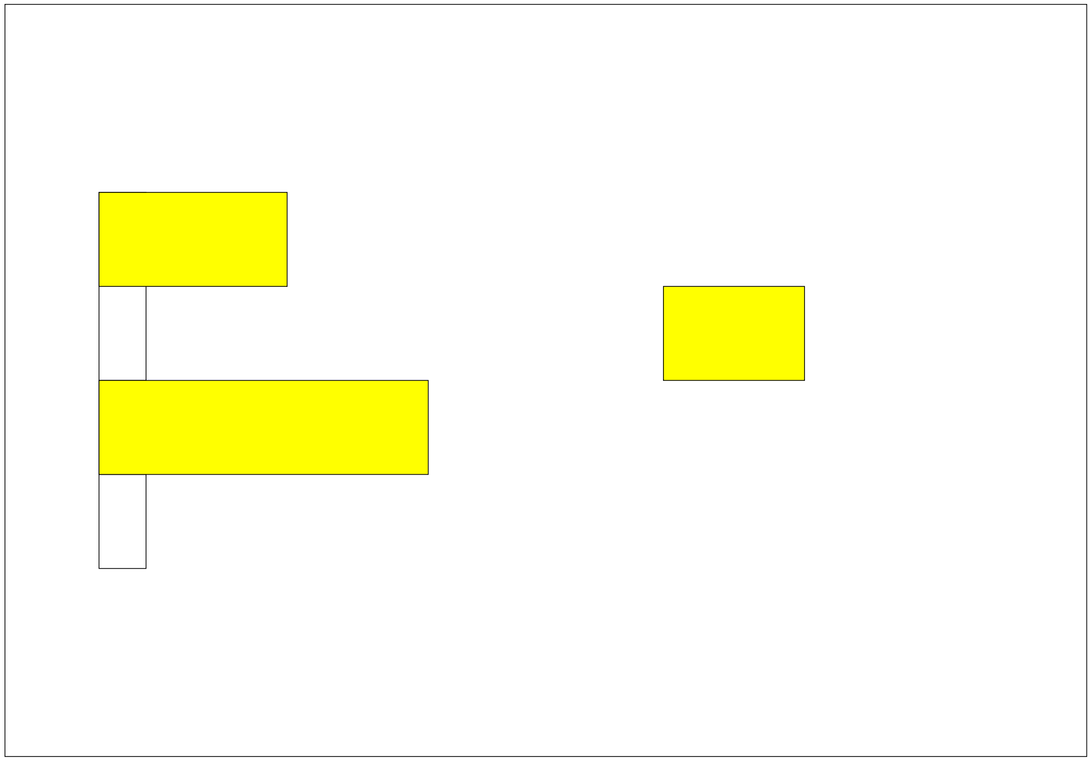
Example



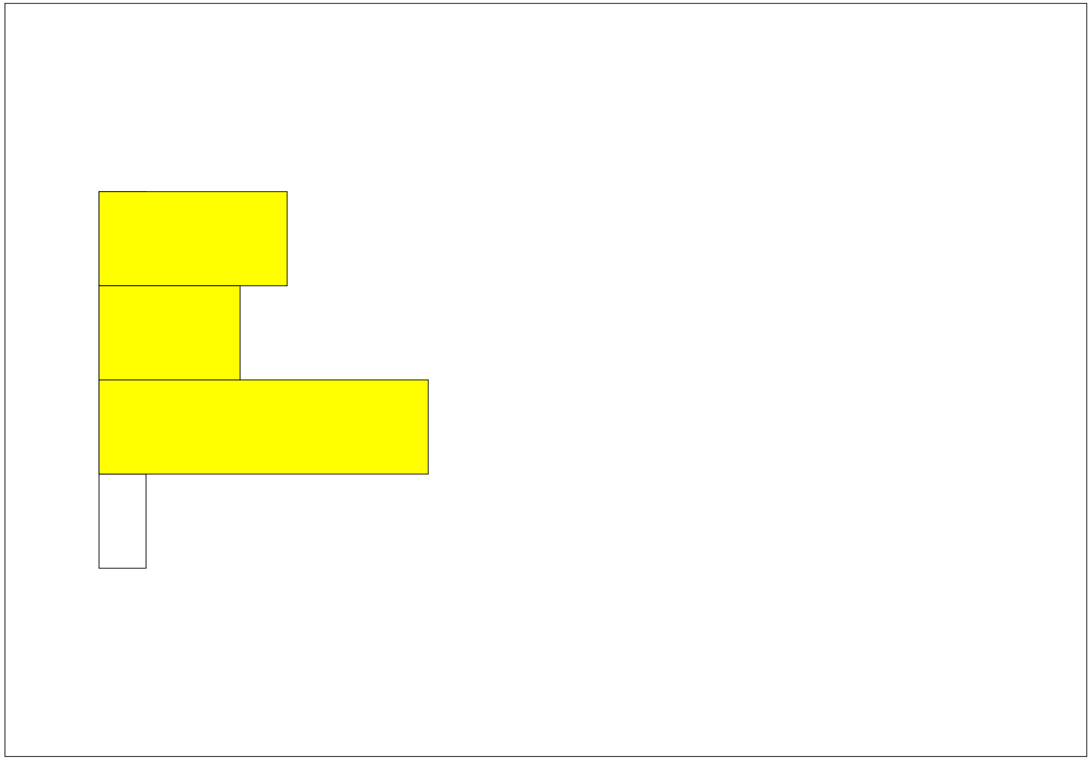
Example



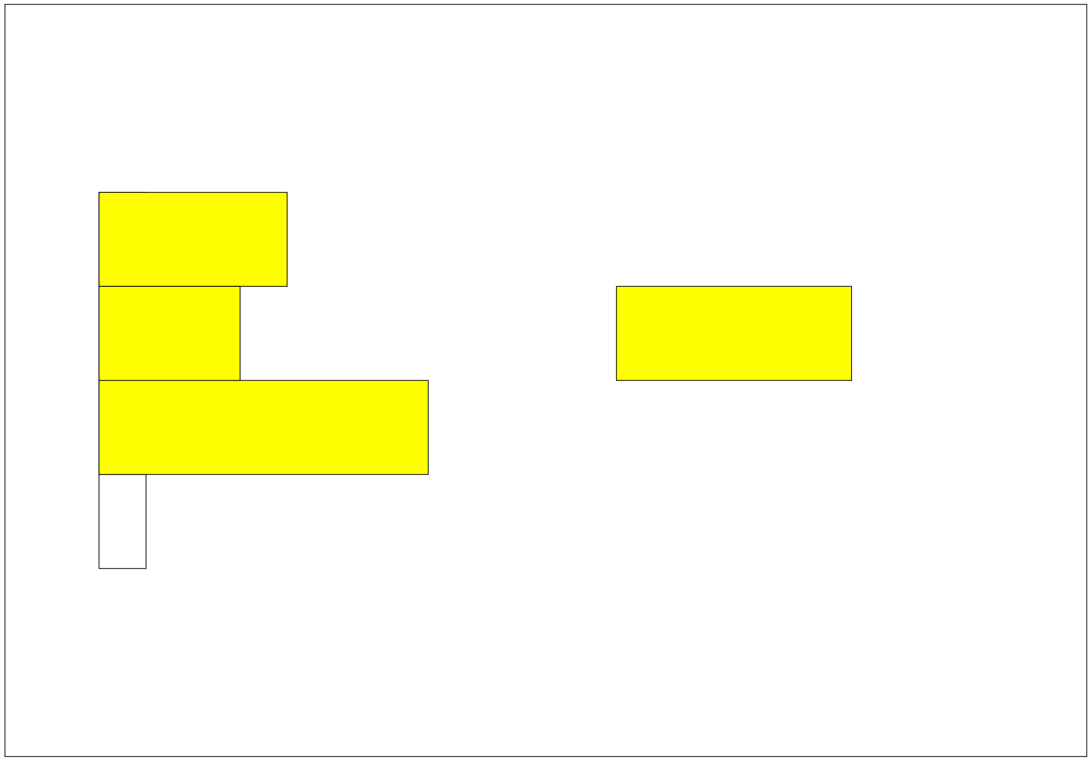
Example



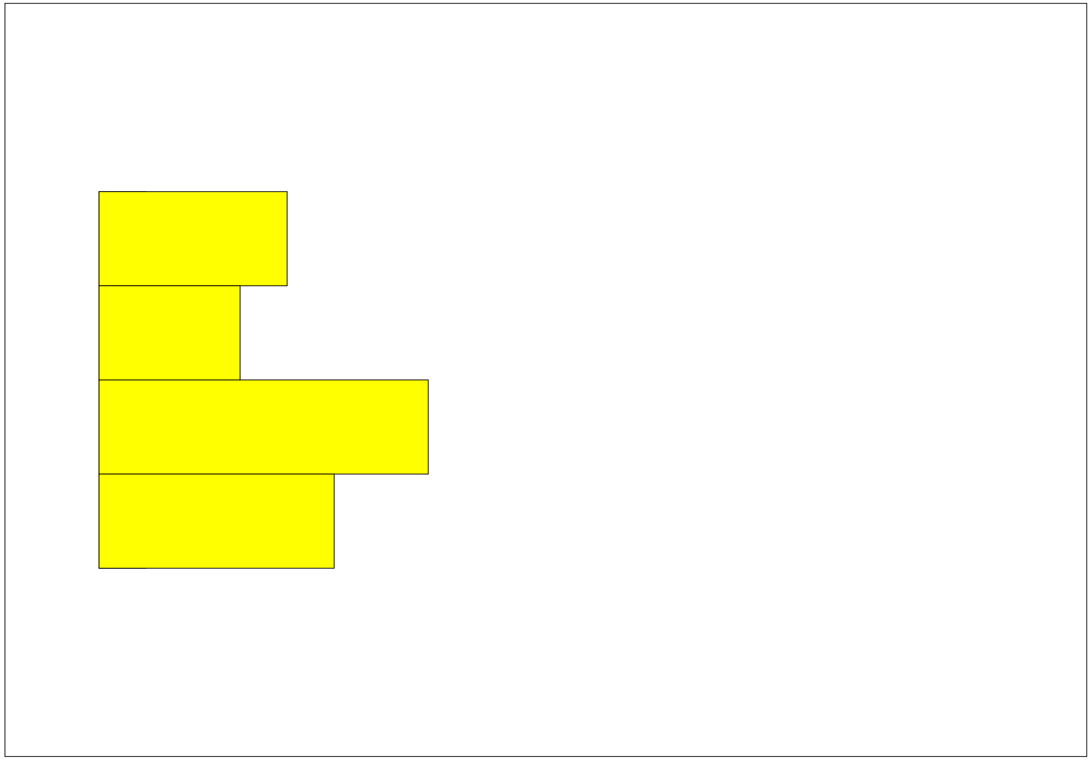
Example



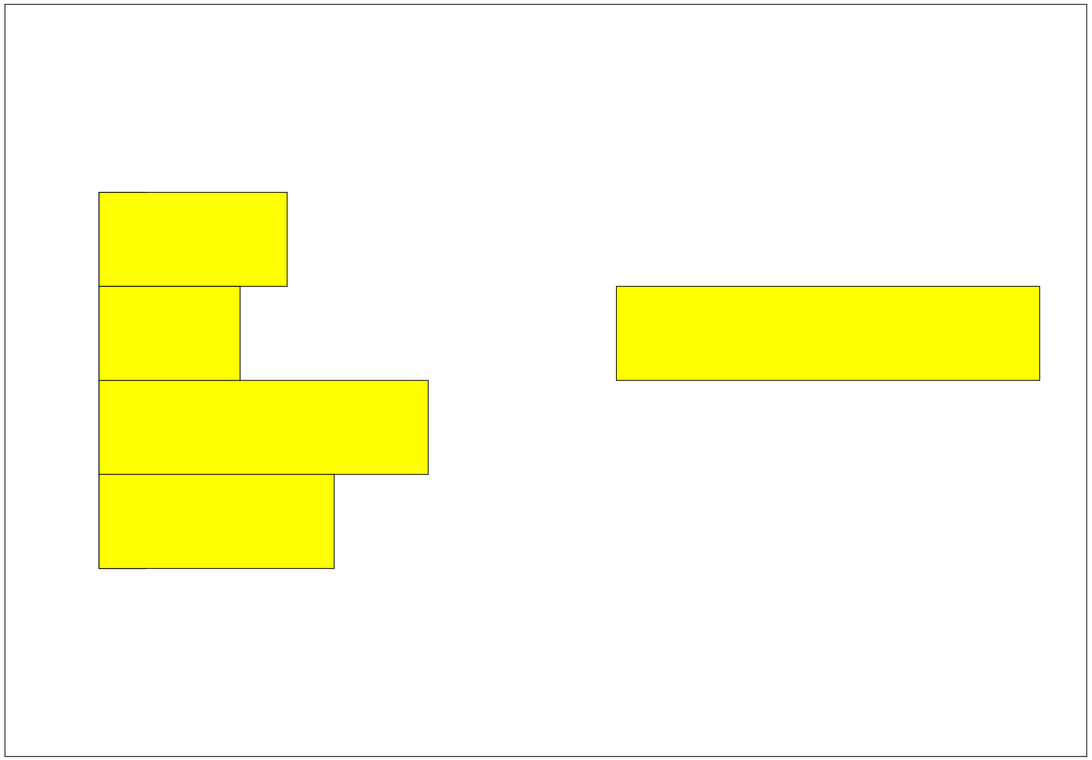
Example



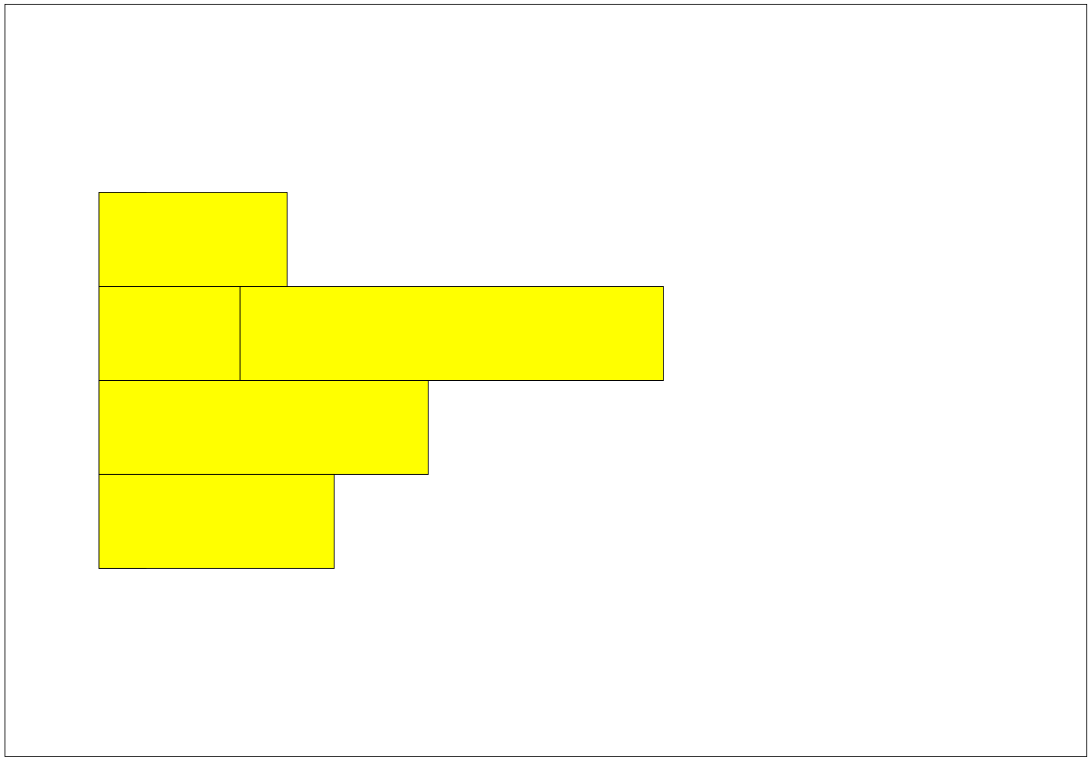
Example



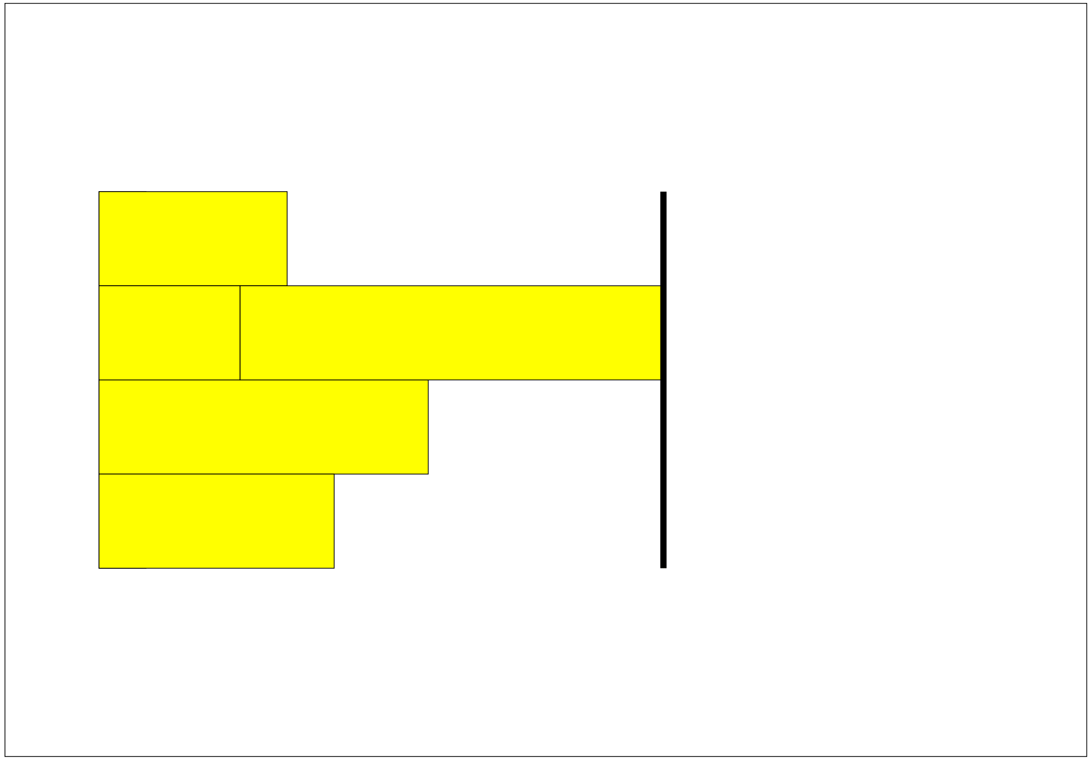
Example



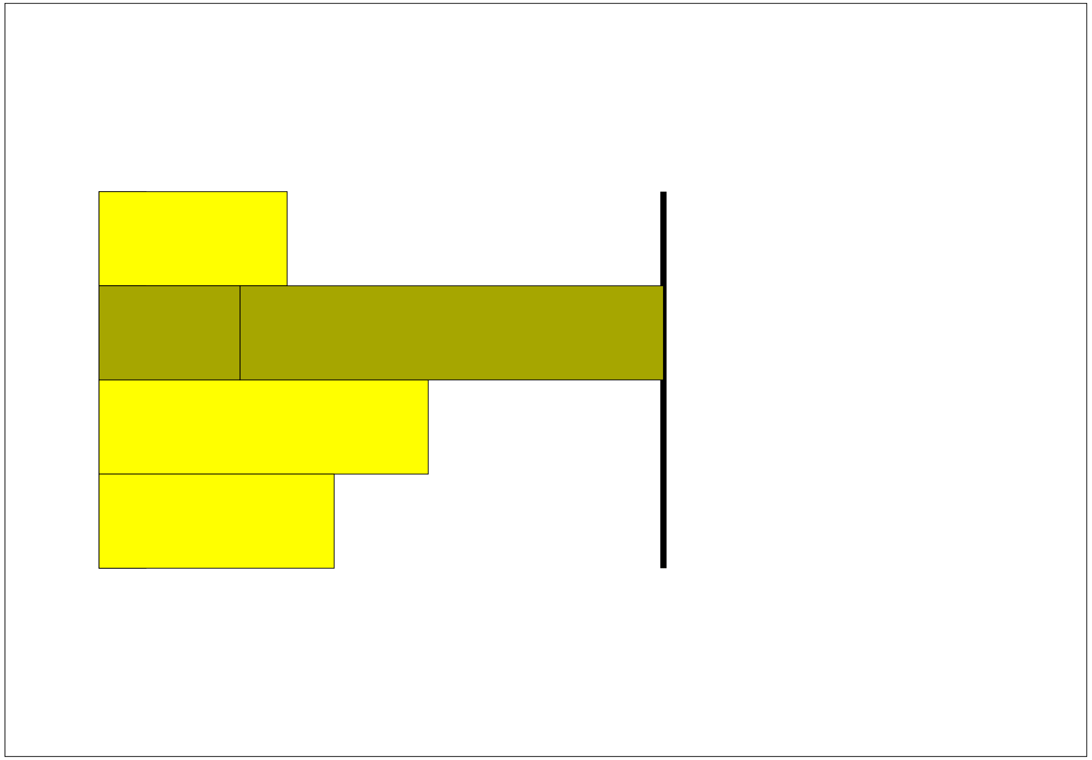
Example



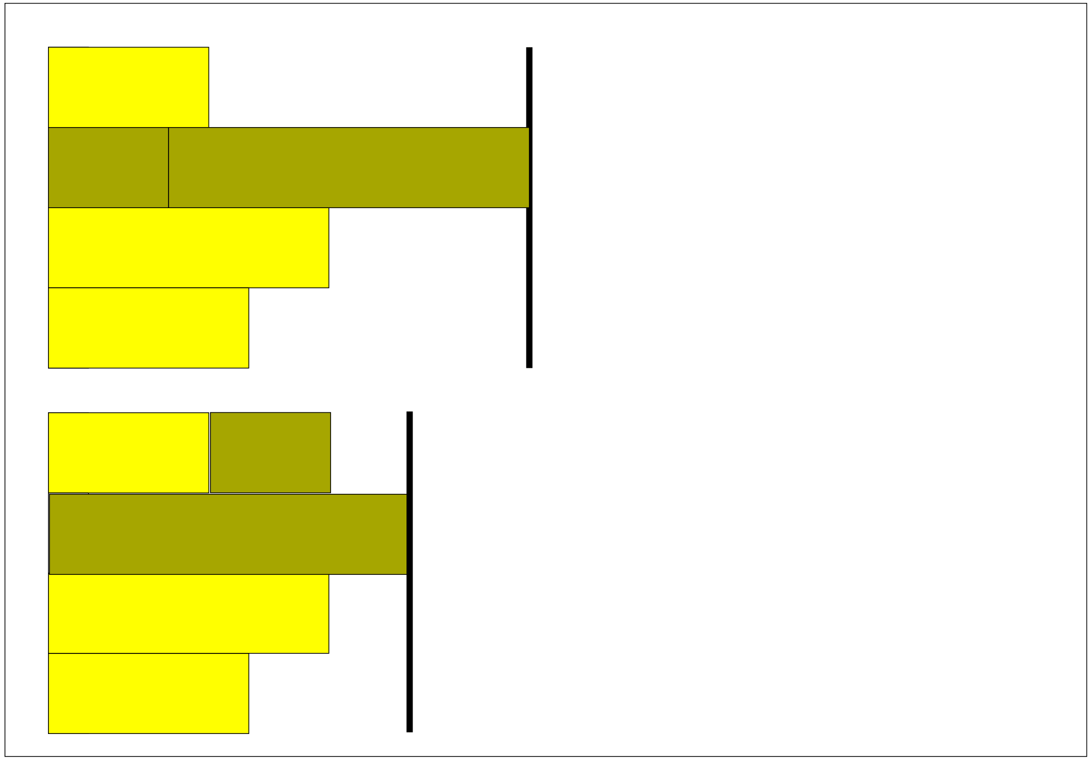
Example



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 machine having least total processing time. (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 and r be the remaining total processing time. Hence $A(\sigma) = r + w$.
- When w was assigned greedily, all other machines also have 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, $A(\sigma) = r + w \leq (2 - \frac{1}{m})OPT(\sigma)$.

Self-organizing lists

- Consider a list L of n elements $\{a_1, a_2, \ldots, a_n\}$.
- Cost of accessing an element x in L is $rank(x)$.
- Algorithm is allowed to reorganize the list using transposition of adjacent elements.
- Cost of a single transposition 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.
- MTF (Move to Front): When an element is accessed, move it to front of the list.
- Hence accessing an element x would incur a cost at most $2 \cdot rank(x)$.

Amortized analysis - Potential functions

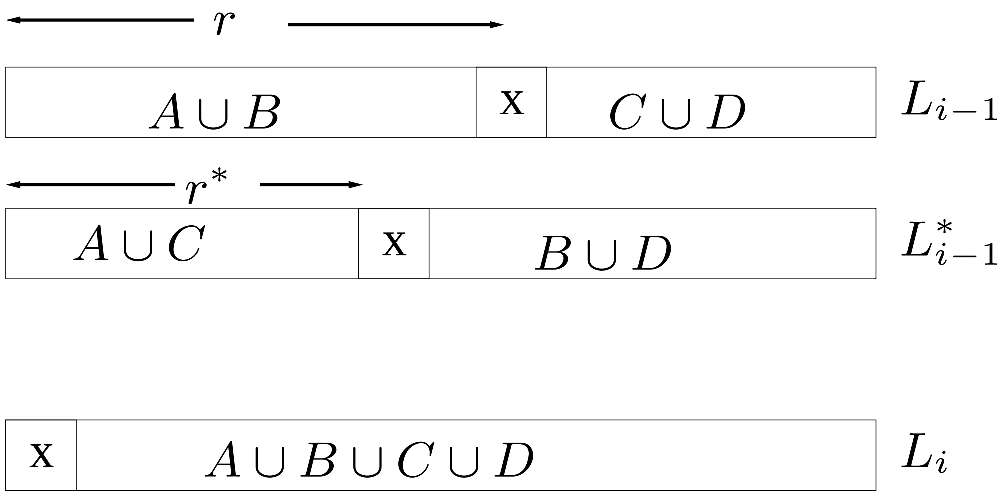
- An aggregate cost analysis technique using potential functions.
- Consider a request sequence σ of length s.
- Let C_i denote the cost of MTF to serve ith request. Let $C(\sigma) = \sum_{i=1}^{s} C_i$.
- Define a potential function Φ_i that maps the state of the list after i rounds to a non negative real number.
- We will define Φ such that $\Phi_0 = 0$.
- Amortized cost of the algorithm in step i denoted by $\hat{C}_i = C_i + \Phi_i - \Phi_{i-1}$.
- Cost of serving σ is
  $C(\sigma) = \sum_{i=1}^{s}(\hat{C}_i + \Phi_{i-1} - \Phi_i) = \Phi_0 - \Phi_s + \sum_{i=1}^{s} \hat{C}_i \leq \sum_{i=1}^{s} \hat{C}_i$

MTF - Amortized analysis

- We will bound cost of serving ith request, say x.
- Lets compare the list configurations L_{i-1} and L^*_{i-1} of MTF and OPT respectively before serving ith request.
- Let $r = rank(x)$ in L_{i-1} and r^* be $rank(x)$ in L^*_{i-1}.
- Hence $C_i \leq 2r$.
- Let C^*_i denote the OPT cost for serving ith request. Let $C^*_i = r^* + t_i$, where t_i is the number of transpositions done by OPT.
- Define Φ_{i-1} to be twice the number of inversions in L_{i-1} with respect to L^*_{i-1}.
- That is Φ_{i-1} is the no. of ordered pairs of elements in L_{i-1} that appear in opposite order in L^*_{i-1}.
- Clearly $\Phi_{i-1} \geq 0$ and $\Phi_0 = 0$. (Both start with same list)

MTF - Amortized analysis

- We now bound $\Phi_i - \Phi_{i-1}$

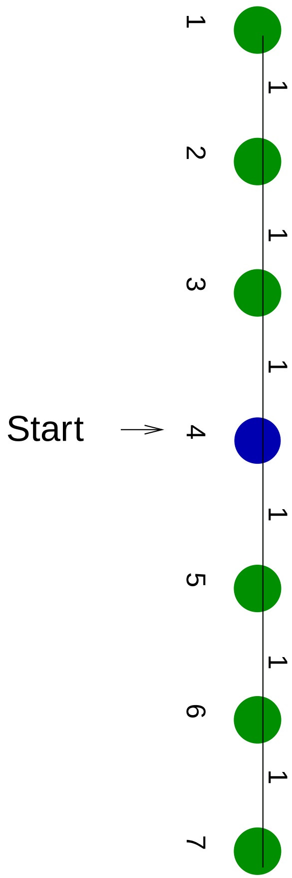


- MTF step creates $|A|$ new inversions and destroys $|B|$ existing inversions.
- Note that $r = |A| + |B| + 1$ and $r^* = |A| + |C| + 1$.
- New inversions due to t_i transpositions of OPT are at most t_i.
- Hence the potential difference $\Delta\Phi$ is,
  $\Phi_i - \Phi_{i-1} \leq 2(|A| - |B| + t_i) \leq 4|A| + 2 + 2t_i - 2r \leq 4C_i^* - 2r$

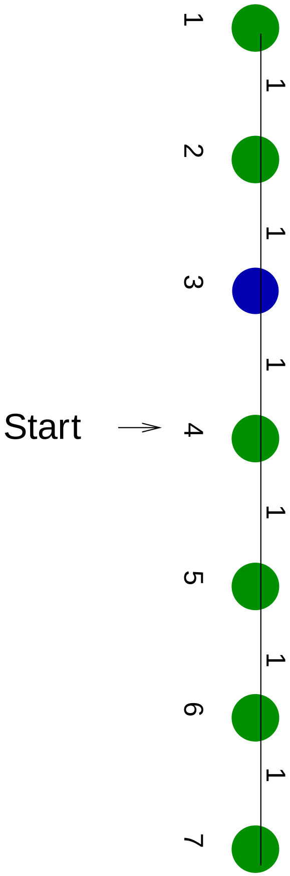
MTF - Amortized analysis

- Hence $\hat{C}_i = C_i + \Phi_i - \Phi_{i-1} \leq 2r + \Phi_i - \Phi_{i-1} \leq 4C_i^*$.
- Since $C(\sigma) \leq \sum_{i=1}^{s} \hat{C}_i$, we obtain $C(\sigma) \leq 4C^*(\sigma)$.
- Thus MTF is 4 competitive.

The Metrical Task Systems Framework



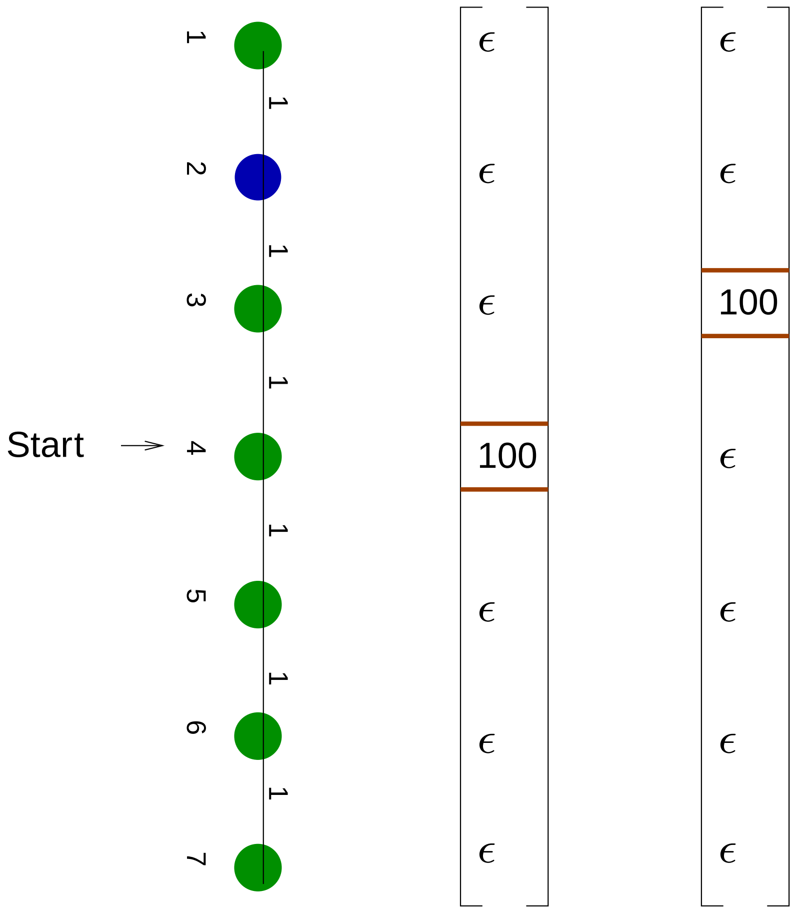
The Metrical Task Systems Framework



$$\left[\begin{array}{c} \epsilon \\ \epsilon \\ \epsilon \\ 100 \\ \epsilon \\ \epsilon \\ \epsilon \end{array}\right]$$

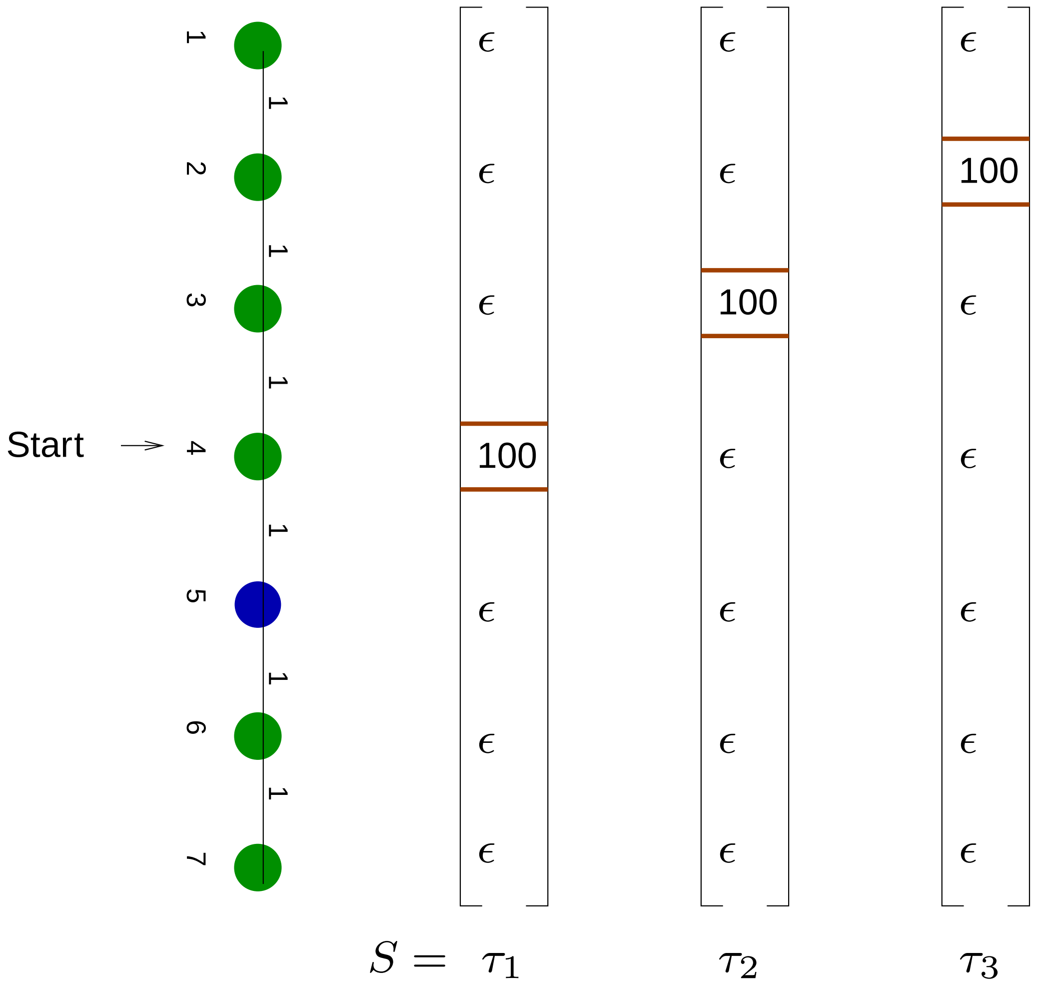
$$S = \tau_1$$

The Metrical Task Systems Framework

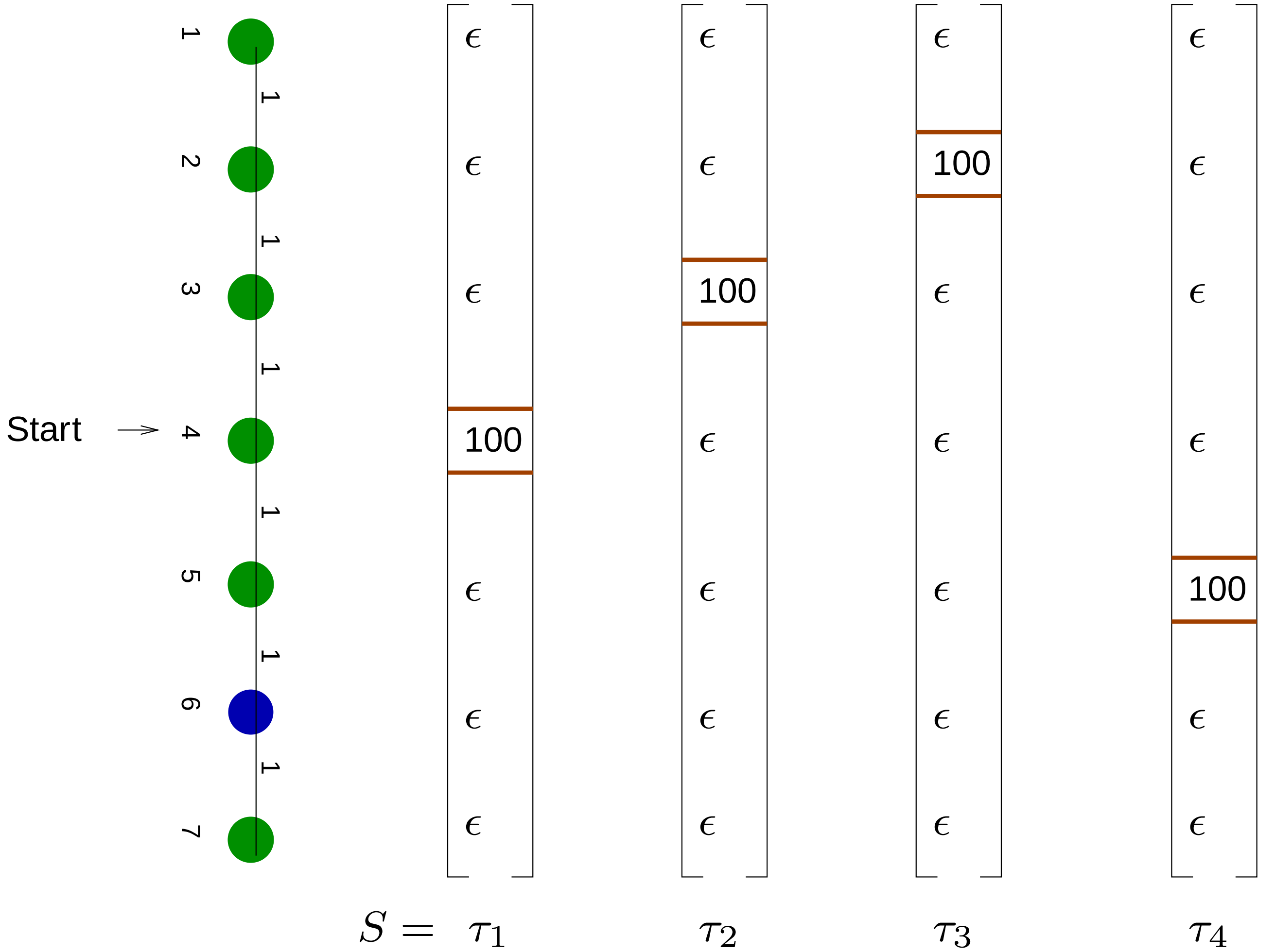


$S = \quad \tau_1 \qquad \tau_2$

The Metrical Task Systems Framework



The Metrical Task Systems Framework



$$ALG[S] = 6 + 4\epsilon, \quad \mathrm{opt}[S] = 2 + 4\epsilon. \quad \mathrm{c.ratio}(S) = \frac{6+4\epsilon}{2+4\epsilon} \approx 3$$

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_i denote the node where A resides after i rounds.
- The adversarial input σ is such that in round t, processing cost at node v_{t-1} is ϵ and 0 everywhere else.

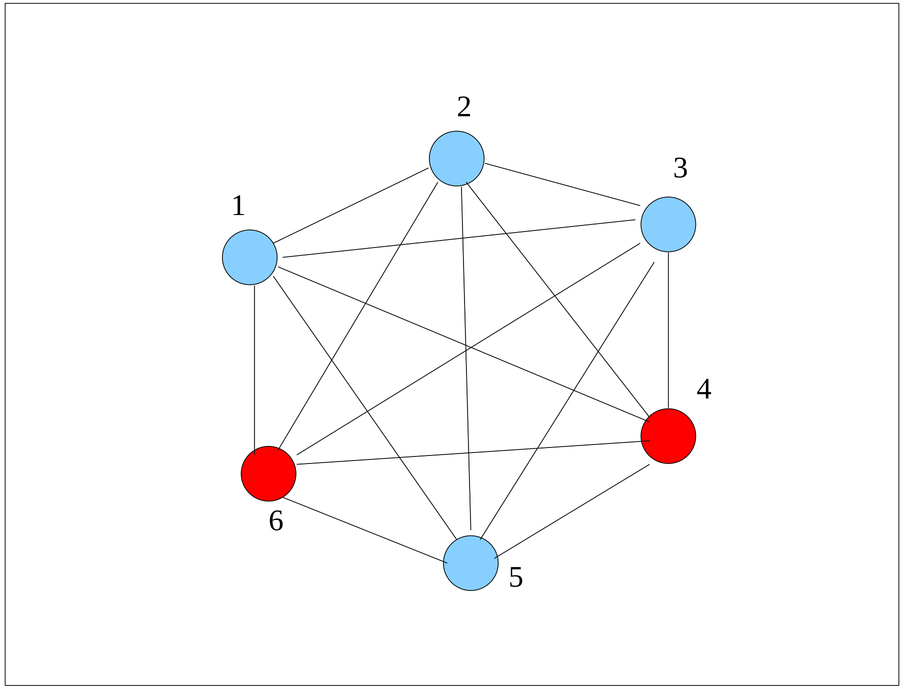
Metrical Task Systems (MTS) – Lower Bound

- 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

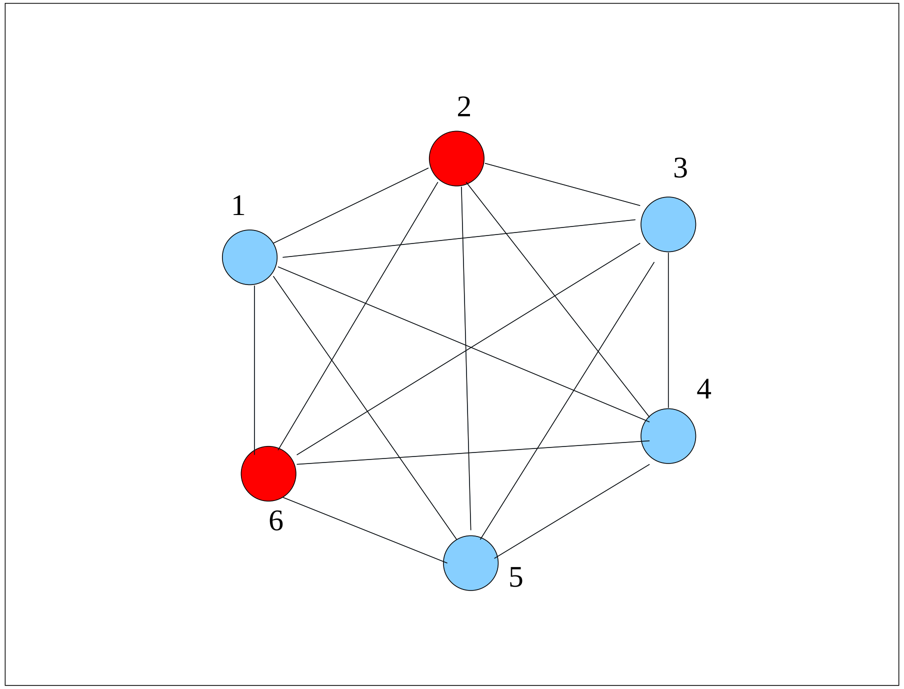
- There are k machines/servers that can move around in an n node weighted graph (which is a metric)
- No two servers reside in the same node.
- Each request $\sigma(i)$ is a node in the graph where the request should be served.

2–server



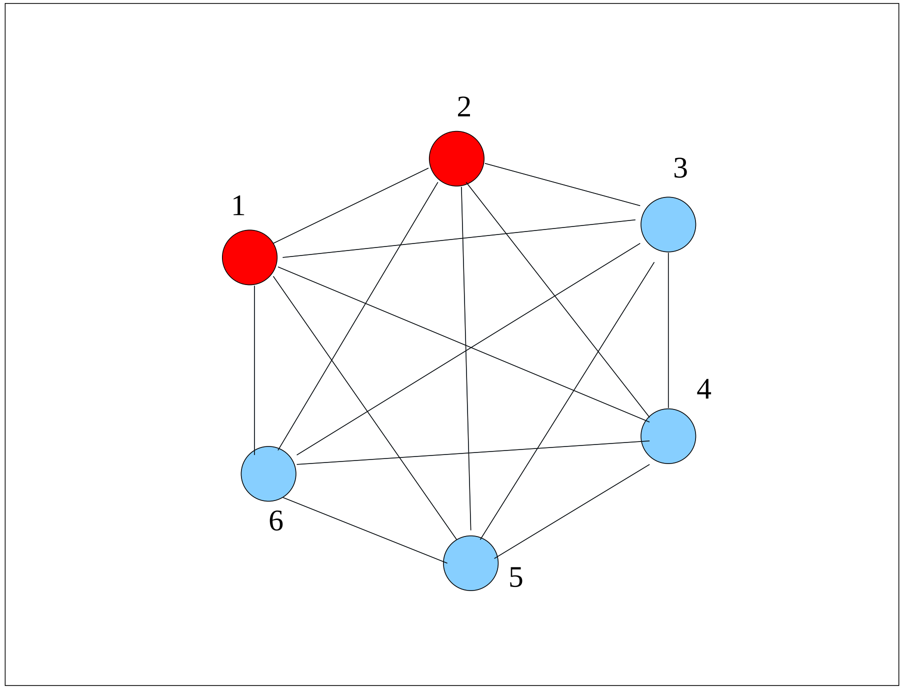
$\sigma =$

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.