1 Introduction

In this lecture we will deal with admission control and routing on general graphs, admission control on the line - lower bounds and classify and randomly select algorithm.

2 Internal Scheduling

**Definition:** Requests are intervals and have bandwidth of 1. Place requests on the line such that no request shall overlap another.

2.1 Preemptive internal scheduling:

- Allowed to dismiss an old request that was previously accepted. Such a request will be annotated **preempted**.
- Not allowed to accept old request that were previously dismissed

**Objective:** Maximize the number of requests accepted.

**Observation:** Preemption may benefit the online algorithm, but it is not useful for the offline OPT.

3 Possible solutions

3.1 Preempt any shorter request

Counter example: A series of lengths \((\sqrt{n}, \sqrt{n} - 1, \sqrt{n} - 2, \ldots)\), each request overlaps by one unit with the previous interval, see figure 1.

The online algorithm is left with the last interval alone, thus \(ON(\sigma) = 1\), and OPT may take all even or odd intervals, resulting in \(OPT(\sigma) = \frac{\sqrt{n}}{2}\).

3.2 Preempt only if the new request is contained in the old one

Counter example shown in figure 2, presenting behavior for the online decisions.

The online algorithm is left with the last interval alone, thus \(ON(\sigma) = 1\), and OPT may takes all the small requests of 2, resulting in \(OPT(\sigma) = \frac{n}{2}\).
\[ t = \sqrt{n} - 1 \]

\[
\begin{array}{c}
\text{...} \\
\text{...} \\
t = 2 & \sqrt{n} - 2 \\
t = 1 & \sqrt{n} - 1 \\
t = 0 & \sqrt{n}
\end{array}
\]

Figure 1: Counter example: A sequence of lengths (\(\sqrt{n}, \sqrt{n} - 1, \sqrt{n} - 2, \ldots\)), each request overlaps by one unit with the previous one.

\[
\begin{array}{c}
t = 5 \\
t = 4 \\
t = 3 \\
t = 2 \\
t = 1 \\
t = 0
\end{array}
\quad
\begin{array}{c}
2(R) \\
n - 10 (A) \\
2(R) \\
n - 6 (P) \\
2(R) \\
n - 2 (P)
\end{array}
\]

Figure 2: A - Request is accepted (at the current time of figure) P - Request was accepted and preempted by another request later R - Request was rejected

\subsection{3.3 the LEVEL HEADED algorithm}

Accept request A if:

- A is over a vacant space
- A contained by another
- A overlaps old requests of at least double the length of A

See figures 3.3 to 5 for examples.

\section{4 Analysis of the LEVEL HEADED algorithm}

\textbf{Theorem}: The LEVEL HEADED algorithm is \(O(\log n)\)
Figure 3: example Request is preempted over double length

Figure 4: example Request is preempted over two double length requests

Figure 5: example Request is not half shorter, and thus rejected

Proof:
At the time request A arrives:

- it may intersect with at most one older request on the right
- it may intersect with at most one older request on the left
- it may contain multiple smaller requests

We create a graph where each request is represented as a node (see figure 6). The graph can be partitioned into separated disjoint trees, where the root of each tree is a request that survived to the end of the process. The request that survived is responsible, either in a direct manner or an indirect, to the rejection and preemption of the other request along the tree. Every node of type A (Accepted) or P (preempted) has at most two sons of type P, and possibly many son nodes of type R (Rejected).
Figure 6: The requests tree: each request is represented by a node. Blue nodes are the requests that survived to the end. If a request did not survive then it was either preempted (represented by a straight edge), or it was rejected (circular edge)
Lemma 1: If request $P$ was preempted (either in a direct manner or an indirect) by request $A$, then the distance between them on the line is equal or smaller to $|P|$, i.e. $d(A, P) \leq |P|$.  
Proof: See figure 7 for the worst case preemption sequence. The maximum distance on the line is $\frac{|P|}{2} + \frac{|P|}{4} + \frac{|P|}{8} + ... \leq |P|$.

$$
\begin{array}{cccccc}
\text{t=final} & 1 & \\
\vdots & & \\
\text{t=2} & & \frac{|P|}{4} & & \vdots \\
\text{t=1} & & \frac{|P|}{2} & & \\
\text{t=0} & |P| & & & & \\
\end{array}
$$

Figure 7: If request $P$ was preempted (either in a direct manner or an indirect) by request $A$, then the distance between them on the line is equal or smaller to $|P|$.

Lemma 2: If request $R$ is in the tree of request $A$ (that is, $R$ was rejected by a request that was directly or indirectly preempted by $A$) then the distance between them on the line is equal or smaller to $4 \times |R|$.  
Proof: See figure 8 for the worst case preemption sequence. The maximum distance on the line is $|P| + \frac{|P|}{2} + \frac{|P|}{4} + \frac{|P|}{8} + ... \leq 2|P| \leq 2 \times 2|R| = 4|R|$.

$$
\begin{array}{cccccc}
\text{t=final} & \text{A} & \\
\vdots & & \\
\text{t=3} & & \frac{|P|}{4} & & \vdots \\
\text{t=2} & & \frac{|P|}{2} & & \\
\text{t=1} & |R| & & & & \\
\text{t=0} |P| \leq 2|R| & & & & \\
\end{array}
$$

Figure 8: If request $R$ was preempted and is under the tree of request $A$, then the distance between them on the line is equal or smaller to $4|R|$.

Lemma 3: Each request $A$ has at most constant number (24) of requests $P$ and $R$ of length $[2^i, 2^{i+1}]$ which OPT can accept, for a given $i$.  
Conclusion: For each request that ON had accepted (of type A), OPT could have received up to $24 \times \log(n)$ requests (there are $\log(n)$ ranges), and hence the algorithm is $O(\log(n))$ competitive.
Proof: If the rejected request L is of length $2^i \leq |L| \leq 2^{i+1}$, then the distance between L and A on the line is at most $4 \cdot 2^{i+1}$ (see figure 9). Note that $|A| \leq 2 \cdot 2^{i+1}$. The total distance on the line from the possible left L to the possible right L is: $2^i \cdot [2 \text{ (given by left L)} + 8 \text{ (the distance between left L and A)} + 4 \text{ (bounded length of A)} + 8 \text{ (the distance between right L and A)} + 2 \text{ (given by right L)}] = 24 \cdot 2^i$. Thus, OPT could have placed at most 24 requests in the range of $[2^i, 2^{i+1}]$ that were rejected as a consequence of A.

<table>
<thead>
<tr>
<th></th>
<th>$4 \cdot 2^{i+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{i+1}$</td>
<td>L</td>
</tr>
<tr>
<td>$L$</td>
<td></td>
</tr>
<tr>
<td>$4 \cdot 2^{i+1}$</td>
<td></td>
</tr>
<tr>
<td>$2^{i+1}$</td>
<td>L</td>
</tr>
</tbody>
</table>

Figure 9: ON could have "lost" at most 24 requests as a consequence of request A.

Remark: Algorithm LEVEL HEADED is $\Omega(\log(n))$ competitive.

Proof: Example in the figure 10 showing the sequence ($\frac{|n|}{2}, \frac{|n|}{4}, \frac{|n|}{8}, \ldots, 2$). ON=1 and OPT takes the even/odd requests, hence $\text{OPT}=\log n$.

$t=\text{final}$
$t=2$
$t=1$
$t=0$

Figure 10: The algorithm is as bad as $\Omega(\log(n))$.

Theorem: Every deterministic preemptive algorithm is $\Omega(\log(n))$ competitive.

Proof: Example in the figure 11 showing the sequence ($2 \cdot \frac{|n|}{2}, 2 \cdot (\frac{|n|}{4} - 1), 2 \cdot (\frac{|n|}{8} - 2), \ldots, 2$), similar to the way we have proved lower bound for restricted machines. ON=1 and $\text{OPT}=\log n$. 

10 - 6
Figure 11: Lower bound for deterministic preemptive algorithm is $\Omega(\log(n))$. 