Lecture 06 - Workflow Optimization
Workflow Representation

- Workflows are DAG’s and thereby we need a graph to represent a workflow.
- Each node in the DAG represents a computation block.
- Each edge in the DAG represent a data flow from one computation block to another.
- Is there any weight associated with the graph?
Workflow Optimization Problem

- Classic Scheduling Problem: How to schedule workflows effectively to the Cloud?
- Is scheduling problem just about mapping tasks in the workflow to machines? No
- Algorithm needs to be provide a workflow schedule based on considering QOS constraints attached to the workflow, execution time, execution cost into consideration.
Workflow Optimization Problem

- **P1**: Given Budget for workflow execution: how to minimize execution time.
- **P2**: Given Deadline for workflow execution: how to minimize execution cost.
- Let us look at P1, which is a budget driven workflow scheduling problem.
A topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge \((u,v)\) from vertex \(u\) to vertex \(v\), \(u\) comes before \(v\) in the ordering.

Let us look at one example:
How to implement the workflow graph representation?

- Classic way of doing is implementing a graph using adjacency matrix. (Remember - Data Structures and Algorithms class)
- New way of doing is using the JGraph Library
- Let us look at some code.
Class Activity Graph 01:

- Try out JGraph (change code) for the following graphs and provide the topological sort in Slack for class participation credits:
Class Activity Graph 02:

- Try out JGraph for the following graphs and provide the topological sort in Slack for class participation credits:
Class Activity Graph 03:

- Try out JGraph for the following graphs and provide the topological sort in Slack for class participation credits:
Class Activity Graph 04:

Try out JGraph for the following graphs and provide the topological sort in Slack for class participation credits:
Workflow Ranking

- **Upward Rank/ Exit Rank:**
  \[
  \begin{cases}
    F_p(T_i), & \text{if } T_i = T_{\text{exit}}, \\
    F_p(T_i) + \max_{T_j \in \text{succ}(T_i)}(F_c(i, j) + F_u(T_j)), & \text{otherwise}.
  \end{cases}
  \]

- **Downward Rank/ Entry Rank:**
  \[
  \begin{cases}
    0, & \text{if } T_i = T_{\text{entry}}, \\
    \max_{T_j \in \text{pred}(T_i)}(F_p(T_j) + F_c(j, i) + F_d(T_j)), & \text{otherwise}.
  \end{cases}
  \]
Workflow Ranking Example

- Sort tasks in non-decreasing order of rank: exit: T1, T3, T2, T4, T5, T7, T10, T6, T8, T9

```
<table>
<thead>
<tr>
<th>Tasks</th>
<th>entry rank</th>
<th>exit rank</th>
<th>overall rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0</td>
<td>344</td>
<td>346</td>
</tr>
<tr>
<td>T2</td>
<td>185</td>
<td>141</td>
<td>326</td>
</tr>
<tr>
<td>T3</td>
<td>188</td>
<td>158</td>
<td>346</td>
</tr>
<tr>
<td>T4</td>
<td>241</td>
<td>85</td>
<td>326</td>
</tr>
<tr>
<td>T5</td>
<td>282</td>
<td>64</td>
<td>346</td>
</tr>
<tr>
<td>T6</td>
<td>274</td>
<td>43</td>
<td>317</td>
</tr>
<tr>
<td>T7</td>
<td>279</td>
<td>47</td>
<td>326</td>
</tr>
<tr>
<td>T8</td>
<td>307</td>
<td>39</td>
<td>346</td>
</tr>
<tr>
<td>T9</td>
<td>326</td>
<td>20</td>
<td>346</td>
</tr>
<tr>
<td>T10</td>
<td>272</td>
<td>46</td>
<td>318</td>
</tr>
</tbody>
</table>
```
Questions