Adaptive Resource Allocation Technique to Stochastic Multimodal Projects: A Distributed Platform Implementation in JAVA

Anabela Tereso (anabelat@dps.uminho.pt)
João Mota (joao_mota@fastmail.fm)
Rui Lameiro (rjl@eurotux.com)

University of Minho – Guimarães – Portugal
Topics
1 – Problem Definition
2 – The DP Model
3 – Example Network
4 – Matlab implementation
5 – JAVA implementation
6 – Data Structures
7 – Class Diagram
8 – Code Optimization
9 – Single Station
10 – Cluster
11 – Results
12 – Accuracy of the solutions
13 – Conclusions and Future Research
1. Problem Definition

- Given a **multimodal** activity network under **stochastic** conditions, we want to optimize the resource allocation to minimize cost

  Optimization via DP
1. Problem Definition

Goal

- Determine the resource allocation vector $X_a$, such that the total expected cost is minimized

$$\min \varepsilon \left\{ \sum x_a W_a + c_L \cdot \max \{ 0, Y_n - T \} \right\}$$

$$x \quad a \in A$$
2. The DP Model

- Process used to select set $F$
  1. Determine the longest path in the network
  2. The activities on the longest path will be the decision variables (set $D$)
  3. The others will be the activities to be fixed (set $F$)

- Resource cost of fixed variables

$$rcf = \varepsilon \sum_{i \in F} x_i W_i = \sum_{i \in F} x_i \varepsilon (W_i)$$
2. The DP Model

- First stage

\[
f_1(s_1|F) = \min \varepsilon \{ x_{[1]}W_{[1]} + rcf + c_L \cdot \varepsilon (U) \}
\]

\[x_{[1]} \in D\]

where

\[U = \max \{ 0, \gamma_{n-T} \}\]

- Next stages

\[
f_k(s_k|F) = \min \varepsilon \{ x_kW_k + \varepsilon f_{k-1}(s_{k-1}|F) \}
\]

\[x_k \in D\]
3. Example Network

D = \{x1, x2\} \quad F = \{x3\}

Longest Path
3. Example Network

<table>
<thead>
<tr>
<th>Activity a</th>
<th>Parameter $\lambda_a$</th>
<th>Expected Work Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.20</td>
<td>5.0</td>
</tr>
<tr>
<td>2</td>
<td>0.10</td>
<td>10.0</td>
</tr>
<tr>
<td>3</td>
<td>0.07</td>
<td>14.29</td>
</tr>
</tbody>
</table>

$T=16$

$C_L=2$

$0.5 \leq x_a \leq 1.5$

Unit: weeks
3. Example Network

- D: discretized in \{0.5, 0.75, 1.0, 1.25, 1.5\} \((k_1=5)\)
- F: discretized in \{0.5, 1.0, 1.5\} \((K_2=3)\)
- W: discretized in 4 values
  
p.e. \(W_2 \sim \exp(0.1) \rightarrow \{1.37, 4.77, 10.0, 23.86\}\)

- \(rcf = x_3 \cdot E(W_3) = x_3 / 0.07\)
3. Example Network

- DP iterations

  - Stage 1

    \[ f_1(t_2 \mid F=\{3\}) = rcf + \min_{x_2} E \{ x_2.W_2 + 2.E(U) \} \]

    \[ U = \max \{ 0, \Upsilon_3 - T \} \]

    \[ \Upsilon_3 = \max \{ t_2 + W_2/x_2, W_3/x_3 \} \]

  - Stage 2

    \[ f_2(t_1=0 \mid F=\{3\}) = \min_{x_1} E \{ x_1.W_1 + E[f_2(\Upsilon_2)] \} \]

    \[ \Upsilon_2 = W_1/x_1 \]
3. Example Network

- Computational Tests
  - Pentium IV E, 3 GHz, 1 GB
  - Windows XP Professional

- Solution
  
  \[(x_1^*, x_3^*) = (1.0, 1.0)\]
  
  with a total expected cost of 43.32
  
  (running time: 0.094 s)
4. Matlab Implementation

- Code dependent on the network topology
- Code composed dynamically
- Code not very structured
- High running times
- Single station implementation
5. JAVA Implementation

- Better running times
- Fast development application
- OO Language
- Code better structured
- Better use of computational resources (dual thread and cluster)
- Different architectures / operating systems
- Open source platform / free of charge
6. Data Structures

- **Node**
  - Immediately preceding / succeeding nodes
  - Activities that connect to the node

- **Activity**
  - Parameter $\lambda$
  - Lower/upper bounds on resource allocation

- **Network**
  - List of activities
  - List of nodes
6. Data Structures

- Resource Combination
  - Object cloned at each stage
  - A clone: a combination of the node’s time values belonging to the present UDC
  - At the end each clone will contain the optimal value of the decision variable, and the expected value of the cost for that stage
7. Class Diagram

**Network**
- activities: Vector
- nodes: Vector
- initialNode: Node
- finalNode: Node

**Node**
- maxTimeLimit: double
- minTimeLimit: double
- times: double
- timesStep: double
- previousNodes: Vector
- succNodes: Vector
- previousActivities: Vector
- succActivities: Vector
- ID: int
- initial: boolean

**UniformlyDirectedCutSet**
- nodes: Vector
- activityInTheLongestPath: Activity
- nodeInTheLongestPath: Node
- nextUDC: UniformlyDirectedCutSet
- fixedActivities: Vector
- nodesOutTheNextUDC: Vector
- nodesInTheNextUDC: Vector
- activitiesToNodeFromNextUDC: HasTable
- previousNodesFromNextUDCNode: HasTable
- initial: boolean

**Problem**
- net: Network
- unitDelayCost: double
- dueDate: double
- workContentAccumulatedProbability: double[]
- workContentProbability: double[]
- udc: Vector
- combinations: long
- done: double

**BestCombination**
- combinations: HasTable
- problem: Problem

**Activity**
- ID: int
- source: Node
- target: Node
- lambda: double
- minResource: double
- maxResource: double
- minDuration: double
- maxDuration: double
- workContent: double[]
- expectedWorkContent: double[]
- resourcePoints: double[]

**ResourceCombination**
- fixedActivitiesResources: HasTable
- timeValues: HasTable
- fixedActivitiesResourcesIndex: HasTable
- fixedValuesIndex: HasTable
- longestPathActivity: Activity
- longestPathActivitiesResource: double
- longestPathActivitiesResourceIndex: int
- problem: Problem
- expectedValue: double
- definedExpectedValue: boolean
8. Code Optimization

- Search method to find the closest combination of node due times
  - Matlab: Sequential search
    - `find(value,array)`
  - JAVA: Hash table (90% faster)
    - HashFunction: Combination $\rightarrow$ Integer
    - $\text{Hash}(\text{combination}) = \sum_{i=1}^{N} T_i \times MTV^i$
9. Single Station

RUN

Better combinations

Combination with total cost

Is it better?

UDC 1

UDC 2

UDC N

Better combinations

Fcombinations

IFORS 2005 - Hawaii
10. Cluster

- Client / server model
- Similar to the SETI@Home project
- 250 resource combinations → client
- End: best solution found
- Communications: TCP/IP sockets
10. Cluster

F combinations

Client

Client

Client

...
### 11. Results

<table>
<thead>
<tr>
<th>Net</th>
<th>n</th>
<th>T</th>
<th>cL</th>
<th>DP Time</th>
<th>DP Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Matlab</td>
<td>Java dual thread</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$k_2 = 3$</td>
<td>$k_2 = 3$</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>16</td>
<td>2</td>
<td>0.094s</td>
<td>0.015s</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>120</td>
<td>8</td>
<td>1.188s</td>
<td>0.140s</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>66</td>
<td>5</td>
<td>4.078s</td>
<td>0.188s</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>105</td>
<td>4</td>
<td>5m 39s</td>
<td>5.218s</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>28</td>
<td>8</td>
<td>10m 08s</td>
<td>22.422s</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>65</td>
<td>5</td>
<td>1h 03m 06s</td>
<td>2m 33s</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>47</td>
<td>4</td>
<td>7h 43m 09s</td>
<td>19m 10s</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
<td>37</td>
<td>3</td>
<td>38h 45m  (\uparrow)</td>
<td>1h 36m 17s</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
<td>188</td>
<td>6</td>
<td>441h 30m  (\uparrow)</td>
<td>18h 16m 56s</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>49</td>
<td>7</td>
<td>117h 40m  (\uparrow)</td>
<td>4h 52m 23s</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>110</td>
<td>10</td>
<td>5285h  (\uparrow)</td>
<td>218h 50m  (\uparrow)</td>
</tr>
</tbody>
</table>
## 12. Accuracy of the Solutions

<table>
<thead>
<tr>
<th>Net</th>
<th>$k_2$</th>
<th>DP Cost</th>
<th>Set of best allocations to the first UDC and $\mathcal{F}$;</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>3</td>
<td>272.298</td>
<td>{1.25, 0.5, 1.5, 0.5, 1.0, 1.0, 1.5, 1.0}</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>266.498</td>
<td>{1.5, 0.5, 1.25, 0.5, 1.25, 1.0, 1.5, 1.0}</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>182.914</td>
<td>{1.25, 1.0, 1.0, 0.5, 1.0, 1.0, 0.5, 1.0, 1.5}</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>172.251</td>
<td>{1.5, 1.0, 0.75, 0.5, 0.75, 0.75, 0.5, 0.75, 1.5}</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>120.335</td>
<td>{1.5, 1.5, 0.5, 0.5, 1.0, 0.5, 1.0, 1.0, 1.0}</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>114.163</td>
<td>{1.5, 1.25, 1.25, 0.5, 0.75, 0.5, 0.75, 1.0, 0.75, 1.0, 0.75}</td>
</tr>
</tbody>
</table>
13. Conclusions and Future Research

- JAVA 25 times faster than Matlab (single machine). No limits for a cluster.
- JAVA code easier to understand.

- Write a new client program to be used in a cluster with computers with light tasks.
- Chunk size dynamically adapted to each client.
- Improve performance by using a small $k_2$, and successively finer mesh.
- Convert AoN to AoA.
References


