Chap 11: Hierarchical Models

Objective: to avoid large models so as to improve solution efficiency.

Ex1:

upper-level model
(a reliability block diagram)

Lower-level model for a bridge
(a reliability graph)
Partial sharpe code shown

\begin{align*}
\text{relgraph} & \quad \text{rbridge} (v1, v2, v3, v4, v5) \\
w & x \quad \exp (v1) \\
x & z \quad \exp (v2) \\
w & y \quad \exp (v4) \\
y & z \quad \exp (v5) \\
\text{bidirect} \\
x & y \quad \exp (v3) \\
\text{end}
\end{align*}

\begin{align*}
\text{block} & \quad \text{rel-in-block} \\
\text{comp} & \quad 11 \quad \exp (u11) \\
\text{comp} & \quad 12 \quad \exp (u12) \\
\text{comp} & \quad 13 \quad \exp (u13) \\
\text{comp} & \quad 14 \quad \exp (u14) \\
\text{comp} & \quad 15 \quad \exp (u15) \\
\text{comp} & \quad \text{bridge1} \quad \text{cdf} (\text{rbridge}; u1, u2, u3, u4, u5) \\
\text{comp} & \quad \text{bridge2} \quad \text{cdf} (\text{rbridge}; u6, u7, u8, u9, u10) \\
\text{parallel} & \quad \text{C} \quad 12 \quad 13 \\
\text{series} & \quad \text{D} \quad \text{bridge1} \quad 11 \quad \text{C} \\
\text{series} & \quad \text{E} \quad 14 \quad \text{bridge2} \quad 15 \\
\text{parallel} & \quad \text{top} \quad \text{D} \quad \text{E} \\
\text{end} \\
\text{eval} (\text{rel-in-block}) & \quad 0 \quad 50000 \quad 500
\end{align*}
Ex2: A queuing model with resource constraints

Within the dashed line is the central server system.

Not in product-form because of resource limitations.

A load dependent center

\[ X(i) = \begin{cases} 
X(i), & \text{if } i = 1,2,\ldots,n \\
X(n), & \text{if } i > n 
\end{cases} \]
* low-level model
  \[
  \text{pfqn inner}(n)  \\
  \begin{array}{lll}
  \text{CPU} & \text{disk1} & P_1 \\
  \text{CPU} & \text{disk2} & P_2 \\
  \text{CPU} & \text{CPU} & 1-P_1P_2 \\
  \text{disk1} & \text{CPU} & 1 \\
  \text{disk2} & \text{CPU} & 1 \\
  \end{array}
  \]
  \[
  \begin{array}{llr}
  \text{CPU} & \text{fcfs} & 1000/20 \\
  \text{disk1} & \text{fcfs} & 1000/30 \\
  \text{disk2} & \text{fcfs} & 1000/42.9 \\
  \end{array}
  \]
  \[
  \begin{array}{ll}
  \text{chain1} & n \\
  \end{array}
  \]

* high-level model
  \[
  \text{pfqn outer}(M)  \\
  \begin{array}{ll}
  \text{term} & 25 \text{ sec.} \\
  \text{term} & 1.0 \\
  \text{term} & 1.0 \\
  \end{array}
  \]

* station types
  \[
  \begin{array}{ll}
  \text{term} & 1/25 \\
  \text{central} & \text{lds} \ X(1), X(2), X(3), X(4) \\
  \end{array}
  \]
  \[
  \begin{array}{ll}
  \text{chain1} & M \\
  \end{array}
  \]

* define function for lds throughput \( X(n) \)
  \[
  \text{func} \quad X(n) = \text{tput}(\text{inner, CPU}; n) \times (1 - P_1 - P_2)
  \]
* can also be obtained as
  \[
  (1000/20 \times \text{util(inner, CPU; n)}) \times (1 - P_1 - P_2)
  \]
* by Little’s Law, i.e., \( x_{\text{CPU}} = \mu_{\text{CPU}} \times \rho_{\text{CPU}} \)

<table>
<thead>
<tr>
<th>Service rate of CPU</th>
<th>Utilization of CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( P_1 = 0.667 )</td>
</tr>
<tr>
<td></td>
<td>( P_2 = 0.233 )</td>
</tr>
</tbody>
</table>

* reporting each terminal user’s response time as
* the number of users (M) increases
  \[
  \begin{array}{ll}
  \text{loop} & i, 0, 4, 1 \\
  \text{expr} & 5 \times (2^i) \\
  \text{expr} & \text{rtime(outer, central; 5 \times (2^i))} \\
  \end{array}
  \]

end
end
Ex3: A queuing model with job priorities

two classes of jobs: 1 & 2

M1 = 3 and M2 = 4

\( \lambda_1 = 1/12; \lambda_2 = 1/7 \)

P0 (class 1) = 1/15
P0 (class 2) = 1/31

P1 (class 1) = 8/15
P1 (class 2) = 5/31

P2 (class 1) = 5/15
P2 (class 2) = 15/31

P3 (class 1) = 1/15
P3 (class 2) = 10/31

class 1: 0.1 sec.
class 2: 0.06 sec.
(class 2 has higher priority at CPU)

disk1: 0.03 sec.
disk2: 0.03 sec.
disk3: 0.03 sec.
* performance measures of interest: response time & queue length at CPU.

* not in product-form because of priority scheduling.

Approximation solution: suppose $u_2$ is the utilization of the CPU dedicated to class 2 jobs. Then the CPU service rate for class 1 jobs is slowed down by a factor of $(1-u_2)$.

* we don’t know $u_2$ since it is an output, but we need it as an input for class 1 jobs.

$\therefore$ use iterative technique

Create two CPUs, one for class 1 & the other for class 2, with the CPU service rate to class 1 jobs reduced by a factor of $(1-u_2)$.
Sharpe code (see p.285, text)

\[
\text{mpfqn} \quad \text{iter (M1, M2, u2)}
\]

* chain 1 for class 1 jobs

chain 1

<table>
<thead>
<tr>
<th>Server</th>
<th>CPU1</th>
<th>disk1</th>
<th>CPU1</th>
<th>disk2</th>
<th>CPU1</th>
<th>disk3</th>
<th>CPU1</th>
<th>terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1</td>
<td>8/15</td>
<td></td>
<td>CPU1</td>
<td>5/15</td>
<td>CPU1</td>
<td>1/15</td>
<td>CPU1</td>
<td>1/15</td>
</tr>
<tr>
<td>disk1</td>
<td>CPU1</td>
<td>1</td>
<td>disk2</td>
<td>CPU1</td>
<td>1</td>
<td></td>
<td>disk3</td>
<td>CPU1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

end

* chain 2 for class 2 jobs

chain 2

<table>
<thead>
<tr>
<th>Server</th>
<th>CPU2</th>
<th>disk1</th>
<th>CPU2</th>
<th>disk2</th>
<th>CPU2</th>
<th>disk3</th>
<th>CPU2</th>
<th>terminals</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU2</td>
<td>5/31</td>
<td></td>
<td>CPU2</td>
<td>15/31</td>
<td>CPU2</td>
<td>10/31</td>
<td>CPU2</td>
<td>1/31</td>
</tr>
<tr>
<td>disk1</td>
<td>CPU2</td>
<td>1</td>
<td>disk2</td>
<td>CPU2</td>
<td>1</td>
<td></td>
<td>disk3</td>
<td>CPU2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

end

end

* Section 2: server types

Section 2

<table>
<thead>
<tr>
<th>Server</th>
<th>CPU1 fcfs</th>
<th>(1-u_2)*1/0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1</td>
<td>1/0.1</td>
<td></td>
</tr>
<tr>
<td>CPU2</td>
<td>1/0.06</td>
<td></td>
</tr>
<tr>
<td>disk1</td>
<td>1/0.03</td>
<td></td>
</tr>
<tr>
<td>disk2</td>
<td>1/0.03</td>
<td></td>
</tr>
<tr>
<td>disk3</td>
<td>1/0.03</td>
<td></td>
</tr>
<tr>
<td>terminals</td>
<td>1/12</td>
<td></td>
</tr>
<tr>
<td>class 1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>class 2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

end

end

Service rate of class 1 jobs is reduced by a factor of (1-u_2)

* Section 3: number of jobs per class

Section 3

<table>
<thead>
<tr>
<th>Class</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

end

end

end
* we don’t know what the initial value of \( u_2 \) is, so make a guess \( u_2 = 0 \) initially

\[ \text{bind } u_2 \text{ mutil (iter, CPU2, 2; 3, 4, 0)} \]

* continue this for a sufficient # of iterations until \( u_2 \) converges ⇒ try 5 times

\[ \text{loop } i, 1, 5, 1 \]

\[ \text{bind } u_2 \text{ mutil (iter, CPU2, 2; 3, 4, u_2)} \]

* outputs are:

<table>
<thead>
<tr>
<th>i</th>
<th>u_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.659839</td>
</tr>
<tr>
<td>2</td>
<td>0.659838</td>
</tr>
<tr>
<td>3</td>
<td>0.659838</td>
</tr>
</tbody>
</table>

(converged after 3 iterations)

* try starting \( u_2 \) with another initial value, say \( u_2 = 0.9 \)

\[ \text{bind } u_2 \text{ 0.9} \]

\[ \text{loop } 1, 1, 5, 1 \]

\[ \text{bind } u_2 \text{ mutil (iter, CPU2, 2; 3, 4, u_2)} \]

* parameters for \( M_1, M_2, \) & \( u_2 \)

\[ \text{M1}=3; \text{ M2}=4 \text{ & } u_2 \text{ is equal to the } u_2 \text{ in the previous iteration} \]

* outputs are:

<table>
<thead>
<tr>
<th>i</th>
<th>u_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.660454</td>
</tr>
<tr>
<td>2</td>
<td>0.659839</td>
</tr>
<tr>
<td>3</td>
<td>0.659838</td>
</tr>
</tbody>
</table>

(3 converged)

\[ \text{print response time & queue size} \]

\[ \text{expr mrtime (iter, CPU1, 1; 3, 4, u_2)} \]

\[ \text{expr mrtime (iter, CPU2, 2; 3, 4, u_2)} \]

\[ \text{mglength (iter, CPU1, 1; 3, 4, u_2)} \]

\[ \text{mglength (iter, CPU2, 2; 3, 4, u_2)} \]

* outputs are:

\[ R_{1, \text{CPU}} = 0.47534 \]

\[ R_{2, \text{CPU}} = 0.10511 \]

\[ n_{1, \text{CPU}} = 1.0911 \]

\[ n_{2, \text{CPU}} = 1.1559 \]

* to be compared with the corresponding parameter values without priority scheduling

\[ R_{1, \text{CPU}} = 0.28483 \]

\[ R_{2, \text{CPU}} = 0.15834 \]

\[ n_{1, \text{CPU}} = 0.76545 \]

\[ n_{2, \text{CPU}} = 1.5197 \]
Ex4: M/M/1/k queue with server failure & repair

P.233, text & p.294

\( \gamma \): failure rate \quad \( \lambda \): job arrival rate

\( \tau \): repair rate \quad \( \mu \): job service rate

1-level model

\[
\begin{align*}
(0,1) & \xrightarrow{\lambda} (1,1) \xrightarrow{\lambda} (2,1) \xrightarrow{\lambda} \ldots \xrightarrow{\lambda} (9,1) \xrightarrow{\lambda} (10,1) \\
(0,0) & \xrightarrow{\lambda} (1,0) \xrightarrow{\lambda} (2,0) \xrightarrow{\lambda} \ldots \xrightarrow{\lambda} (9,0) \xrightarrow{\lambda} (10,0)
\end{align*}
\]

State representation \((a, b)\)

\# of jobs \{ 1 alive, 0 failed \}

\[
\text{prob \{idle server\}} = \text{prob}(0,0) + \text{prob}(0,1)
\]

rejection rate = \text{prob}(10,0) + \text{prob}(10,1)
Two-level model

observation: job arrivals/services are much faster than server failures/repairs

∴ the assumption below is justified:

“the set of states $0,1$, $1,1$ .............. $9,1$, $10,1$ whose transitions are job arrivals and departures will reach equilibrium between the times when a failure/repair occurs.”

⇒ isolate out the fast recurrent set of states from the 1-level model, analyze it for steady-state probabilities & replace it by a single state in the original model.
High-level:

\[ \text{Prob\{idle server\}} = \text{prob}\text{(high-model, 0,0)} + \text{prob}\text{(high-model, 1,0)} \times \text{prob}\text{(low-model, 0,1)} \]

Rejection rate = \text{prob}\text{(high-model, 10,0)} + \text{prob}\text{(high-model, 1,1)} \times \text{prob}\text{(low-model, 10,1)}