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

```
relgraph  rbridge (v1, v2, v3, v4, v5)
w  x  exp (v1)
x  z  exp (v2)
w  y  exp (v4)
y  z  exp (v5)
bidirect
x  y  exp (v3)
end
block  rel-in-block
comp  11  exp (u11)
comp  12  exp (u12)
comp  13  exp (u13)
comp  14  exp (u14)
comp  15  exp (u15)
comp  bridge1  cdf (rbridge; u1, u2, u3, u4, u5)
comp  bridge2  cdf (rbridge; u6, u7, u8, u9, u10)
parallel  C  12  13
series  D  bridge1  11  C
series  E  14  bridge2  15
parallel  top  D  E
end
eval  (rel-in-block)  0  50000  500
end```

Ex2: A queuing model with resource constraints

P.277

# of running jobs in the central server is limited to n

Within the dashed line is the central server system

Not in product-form because of resource limitations

In product-form: both servers can be evaluated independently

$$X(i) = \begin{cases} X(i), & \text{if } i = 1, 2, \ldots, n \\ X(n), & \text{if } i > n \end{cases}$$
* low-level model

\[ \text{pfqn} \quad \text{inner}(n) \]

CPU disk1 P1
CPU disk2 P2
CPU CPU 1-P1P2
disk1 CPU 1
disk2 CPU 1
end

CPU fcfs 1000/20
disk1 fcfs 1000/30
disk2 fcfs 1000/42.9
end

chain1 n
central (lds)
end

* high-level model

\[ \text{pfqn} \quad \text{outer}(M) \]

term central 1.0
central term 1.0
end

* station types

term is 1/25
central lds X(1), X(2), X(3), X(4)
end

chain1 M
end

* define function for lds throughput X(n)

\[ \text{func} \quad X(n) \quad \text{tput(\text{inner},CPU;n)}*(1- P1-P2) \]

* can also be obtained as

\[ (1000/20 * \text{util(\text{inner},CPU;n}))*(1- P1-P2) \]

* by Little’s Law, i.e., \( x_{CPU} = \mu_{CPU} * \rho_{CPU} \)

bind
P1 0.667
P2 0.233
end

* reporting each terminal user’s response time in
* the central system as the number of users (M)
* increases

loop i, 0, 4, 1
expr 5*(2^i)
expr rtime(outer, central; 5*(2^i))
end
end

\[ \begin{align*}
\text{Service} & \quad \text{rate of} \\
\text{Utilization} & \quad \text{CPU} \\
\text{CPU} & \quad \text{Utilization} \\
\mu_{CPU} & \quad \rho_{CPU}
\end{align*} \]
Ex3: A queuing model with job priorities

two classes of jobs: 1 & 2

low priority high priority at the CPU only

\[ \lambda_1 = \frac{1}{12}; \lambda_2 = \frac{1}{7} \]

\[ P_0(\text{class 1}) = \frac{1}{15} \]
\[ P_0(\text{class 2}) = \frac{1}{31} \]

\[ P_1(\text{class 1}) = \frac{8}{15} \]
\[ P_1(\text{class 2}) = \frac{5}{31} \]

\[ P_2(\text{class 1}) = \frac{5}{15} \]
\[ P_2(\text{class 2}) = \frac{15}{31} \]

\[ P_3(\text{class 1}) = \frac{1}{15} \]
\[ P_3(\text{class 2}) = \frac{10}{31} \]

\[ M_1 = 3 & M_2 = 4 \]

\[ \text{class 1: 0.1 sec.} \]
\[ \text{class 2: 0.06 sec.} \]

(class 2 has higher priority at CPU)
* 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.

∴ 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)

mpfqn iter (M1, M2, u2)

* chain 1 for class 1 jobs
chain 1
CPU1 disk1 8/15
CPU1 disk2 5/15
CPU1 disk3 1/15
CPU1 terminals 1/15
disk1 CPU1 1
disk2 CPU1 1
disk3 CPU1 1
end

* chain 2 for class 2 jobs
chain 2
CPU2 disk1 5/31
CPU2 disk2 15/31
CPU2 disk3 10/31
CPU2 terminals 1/31
disk1 CPU2 1
disk2 CPU2 1
disk3 CPU2 1
terminals CPU2 1
end

Section 2: server types

CPU1 fcfs (1-u2)*1/0.1
end
CPU2 fcfs 1/0.06
end
disk1 fcfs 1/0.03
end
disk2 fcfs 1/0.03
end
disk3 fcfs 1/0.03
end

terminals fcfs 1/0.03
end

Service rate of class 1 jobs is reduced by a factor of (1-u2)

Section 3: number of jobs per class

1 M1
2 M2
end

Section 1: routing prob. per class

* Section 2

class 1 class 2
1 1/12
2 1/7
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 \( \Rightarrow \) try 5 times

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

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

outputs are:

\[
\begin{align*}
\text{i=1 } & u_2 \leftarrow 0.659839 \\
\text{i=2 } & u_2 \leftarrow 0.659838 \\
\text{i=3 } & u_2 \leftarrow 0.659838 \\
\end{align*}
\]

(converged after 3 iterations)

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

\[
\text{bind } u_2 \text{ 0.9} \\
\text{loop } 1, 1, 5, 1 \\
\text{bind } u_2 \text{ mutil (iter, CPU2, 2; 3, 4, } u_2) \\
\text{end}
\]

M1=3; M2=4 & \( u_2 \) is equal to the \( u_2 \) in the previous iteration

\[
\begin{align*}
\text{system name } & \text{station name chain 2} \\
\text{parameters for } & \text{M1, M2, & } u_2 \\
\text{R}_1,CPU=0.47534 & \text{R}_2,CPU=0.10511 \\
\text{n}_1,CPU=1.0911 & \text{n}_2,CPU=1.1559 \\
\text{to be compared with the corresponding parameter values without priority scheduling} \\
\text{R}_1,CPU=0.28483 & \text{R}_2,CPU=0.15834 \\
\text{n}_1,CPU=0.76545 & \text{n}_2,CPU=1.5197
\end{align*}
\]
Ex4: M/M/1/k queue with server failure & repair

P.233, text & p.294

\[ \gamma: \text{failure rate} \quad \lambda: \text{job arrival rate} \]
\[ \tau: \text{repair rate} \quad \mu: \text{job service rate} \]

1-level model

\[
\begin{align*}
0,1 & \overset{\lambda}{\longrightarrow} 1,1 & \overset{\lambda}{\longrightarrow} 2,1 & \overset{\lambda}{\longrightarrow} \ldots & \overset{\lambda}{\longrightarrow} 9,1 & \overset{\lambda}{\longrightarrow} 10,1 \\
0,0 & \overset{\lambda}{\longrightarrow} 1,0 & \overset{\lambda}{\longrightarrow} 2,0 & \overset{\lambda}{\longrightarrow} 9,0 & \overset{\lambda}{\longrightarrow} 10,0 \\
\tau & \overset{\gamma}{\downarrow} & \tau & \overset{\gamma}{\downarrow} & \tau & \overset{\gamma}{\downarrow} & \tau & \overset{\gamma}{\downarrow} & \tau & \overset{\gamma}{\downarrow} \\
\mu & \overset{\mu}{\downarrow} & \mu & \overset{\mu}{\downarrow} & \mu & \overset{\mu}{\downarrow} & \mu & \overset{\mu}{\downarrow} & \mu & \overset{\mu}{\downarrow} \\
0,0 & \overset{\lambda}{\longrightarrow} 1,0 & \overset{\lambda}{\longrightarrow} 2,0 & \overset{\lambda}{\longrightarrow} 9,0 & \overset{\lambda}{\longrightarrow} 10,0 \\
\end{align*}
\]

State representation \((a, b)\)

\[ \begin{cases} 1 \text{ alive} \\ 0 \text{ failed} \end{cases} \]

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

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

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

\[ \Rightarrow \text{the assumption below is justified:} \]

“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.”

\[ \Rightarrow \text{isolate out the fast recurrent set of states from the 1-level model, analyze it for steady-state probabilities & replace it by a single state in the original model.} \]
High-level:

Low-level:

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

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