A STAFFING ALGORITHM FOR CALL CENTERS WITH SKILL-BASED ROUTING:

SUPPLEMENTARY MATERIAL

by

Rodney B. Wallace

IBM and
The George Washington University
rodney.wallace@us.ibm.com

Ward Whitt

Columbia University
ward.whitt@columbia.edu

August 20, 2004; Revision; March 31, 2005

Abstract

This Internet Supplement provides material omitted from the main paper. In particular, additional details are provided for the three simulation experiments in Sections 4, 6 and 7 in the main paper.

Abstract in Main Paper

Call centers usually handle several types of calls, but it usually is not possible or cost-effective to have every agent be able to handle every type of call. Thus, the agents tend to have different skills, in different combinations. In such an environment, it is challenging to route calls effectively and determine the staff requirements. This paper addresses *both* these routing and staffing problems by exploiting limited cross-training. Consistent with the literature on flexible manufacturing, we find that minimal flexibility can provide great benefits: Simulation experiments show that when (i) the service-time distribution does not depend on the call type or the agent and (ii) each agent has only two skills, in appropriate combinations, the performance is almost as good as when each agent has all skills. We apply this flexibility property to develop an algorithm for both routing and staffing, aiming to minimize the total staff subject to per-class performance constraints. With appropriate flexibility, it suffices to use a suboptimal routing algorithm. Simulation experiments show that the overall procedure can be remarkably effective: The required staff with limited cross training can be nearly the same as if all agents had all skills. Hence the overall algorithm is nearly optimal for that scenario.

1. Introduction

This Internet Supplement provides material omitted from the main paper. In particular, additional details are provided for the three simulation experiments in Sections 4, 6 and 7 in the main paper. Those sections are repeated here (with additions and modifications) as Sections 2, 3 and 4, respectively. In each case the extra details are added. Some of the added material appears in two appendices, with pointers in the appropriate section.

2. The Resource Pooling Experiment

In this section we describe a simulation experiment conducted to investigate the extent to which resource pooling holds in SBR call centers. In particular, we investigate how many skills agents need in order for the performance to be nearly the same as if all agents had all skills. Under our assumptions so far, that universal-agent case is approximately equivalent to the $M/M/C/K$ model with the FCFS discipline, in which there is a single call type and a single queue. However, even when all calls have the same service-time distribution, as we are assuming, these systems are not quite equivalent, because the FCFS discipline associated with the $M/M/C/K$ model is not operating overall in our SBR system (even though it is within each queue).

We consider a call center serving 6 call types and see what happens when all agents possess m skills, allowing m to range from 1 to 6 in separate simulation runs. The model we consider is a *balanced* $M_6/M/90/30$ SBR call center. There are 90 agents and 30 extra waiting spaces. There are 6 call types and thus 6 work groups. The number of agents per work group is $90/6 = 15$; i.e., $C_1 = C_2 = \cdots = C_6 = 15$. The service times are IID exponential random variables with mean 10 minutes, i.e., $1/\mu_1 = \cdots = 1/\mu_6 = 10$ minutes. The number of skills per agent varies from 1 to 6 in different runs.

Since the number of agents is 90, each work group can have exactly 15 agents, and, when $m \geq 2$, there will be exactly $15/5 = 3$ agents with each of the $6 \times 5 = 30$ combinations of the possible primary and secondary skills. When $m > 2$, we do not try hard to optimally balance the skills at lower priority levels. Instead, we let each successive skill beyond the skill assigned at priority level 2 be the next available skill. For example, suppose that, when $m = 2$, row i of the agent-skill matrix A is $(5, 3, 0, 0, 0, 0)$. When we increase the number of skills per agent, row i is changed successively to $(5, 3, 4, 0, 0, 0)$, $(5, 3, 4, 6, 0, 0)$, $(5, 3, 4, 6, 1, 0)$ and $(5, 3, 4, 6, 1, 2)$. This procedure guarantees that each skill appears the same number of times at

each priority level. This procedure also yields appropriate sharing, but evidently not optimal sharing.

However, the full procedure above does not matter much, because we are primarily interested in comparing the cases $m = 1$, $m = 2$ and $m = 6$. When $m = 6$, all agents have all skills, and the call center behaves much like a single-group call center. In contrast, when $m = 1$, the call center behaves much like 6 separate call centers, for which the performance is much worse. When $m = 1$, the call center does not behave *exactly* like separate call centers because of the finite waiting room. Here, when $m = 1$, the six call types share the common waiting room. The shared waiting room leads to much better performance than if the waiting room too were divided into segregated portions without any sharing.

Here is the main point: We show that the performance for $2 \leq m \leq 5$ is nearly the same as for $m = 6$, being much better than when $m = 1$. We thus see that significant resource pooling occurs even when each agent has only two skills. The performance is somewhat better if $m = 3$ than if $m = 2$, but most of the resource-pooling benefit occurs when $m = 2$.

Recall that the offered load with common mean service times is the arrival rate times the mean service time. We consider three different offered loads: 84.0 (normal load), 77.4 (light load) and 90.0 (heavy load). The corresponding traffic intensities are: $84.0/90 = 0.933$ (normal load), $77.4/90 = 0.86$ (light load) and $90/90 = 1.00$ (heavy load). As discussed in Whitt (1992), the interpretation of traffic intensities depends on the number of servers, with the normal load increasing as the number of servers increases. The finite waiting room makes it possible to have traffic intensities greater than 1 (as would customer abandonment, which we are not considering).

In our resource-pooling simulation experiment, we do all possible cases, considering each number of skills with each loading. Thus we perform $6 \times 3 = 18$ simulation runs in all. Each simulation run is based on approximately $800,000$ arrivals, starting after an initial warmup period to allow the system to reach steady state. The warmup period was chosen to correspond to 2000 mean service times, which constituted about 20%–24% of each run. In order to calculate confidence intervals, we used the technique of batch means, dividing each run into 20 batches.

It is important to validate the simulation tool. As described in Chapter 4 of Wallace (2004), the simulation tool was validated by making comparisons with alternative ways for obtaining numerical results. In particular, as summarized in Table 4.1 of Wallace (2004), the simulation was compared with other methods for treating several different special cases. Among these were exact numerical results for the $M/M/C/K$ model and the bilingual call center analyzed by

Stanford and Grassman (1993, 1998). For more complicated models, a comparison was made with Ridley's (2003) simulator for call centers with time-dependent arrival rates, specialized to the case of constant arrival rates.

We show the simulation results in Figure 1. Figure 1 shows 9 individual graphs in a 3×3 arrangement. The columns correspond to the normal, light and heavy load cases, respectively. The rows show estimates of (i) the steady-state blocking probability, (ii) the conditional expected steady-state delay (before beginning service), given that the call enters (is not blocked), and (iii) and the conditional steady-state probability that the delay exceeds 0.5 minutes, given that the call enters. Both the blocking probability and the delay probability are in percentages. For each of the offered-load scenarios, we make 6 different simulation runs, one for each different number of skills. The horizonal axis for each graph specifies the number of skills that the agents have. The full model requires specifying the agent-skill matrix, which we have done above. The six agent-skill matrices themselves are displayed in Appendix A.

Figure 1 dramatically shows the resource pooling. With only one exception, there are significant improvements in performance when agents are given at least two skills. That one exception is the conditional waiting-time tail probability under heavy loading. However, in all cases we see that most of the benefit is achieved by adding the second skill. Only modest further improvements are achieved when additional skills are provided. Indeed, Figure 1 shows that near-full resource pooling is achieved by giving agents only two skills.

We now give more detailed descriptions of system performance in the 18 cases. These details are provided by Tables 1, 2 and 3. These tables show that the variability of the estimates is not great, because there is relatively little fluctuation in the per-class estimates. The exact values must be equal because the model is symmetric.

In order to put the simulation results into perspective, it is useful to analyze related cases for the $M/M/C/K$ model analytically. We do so in Table 4. First, when $m = 6$, the model is approximately equivalent to the $M/M/90/30$ model. As noted above, when $m = 1$, the model does not reduce to six separate $M/M/15/5$ models, because the six call types actually share the common waiting room of size 30. However, it is clear that the $M/M/15/5$ model is a worst-case bound for the blocking probabilities. On the other hand, the $M/M/15/30$ model is a best-case bound for the blocking probabilities. To put the simulation results for $m = 1$ in perspective, we thus calculate performance measures for the $M/M/15/5$ and $M/M/15/30$ models, with the same arrival and service rates.

From Table 4, we see that indeed the performance in the SBR model with 2 skills is close to

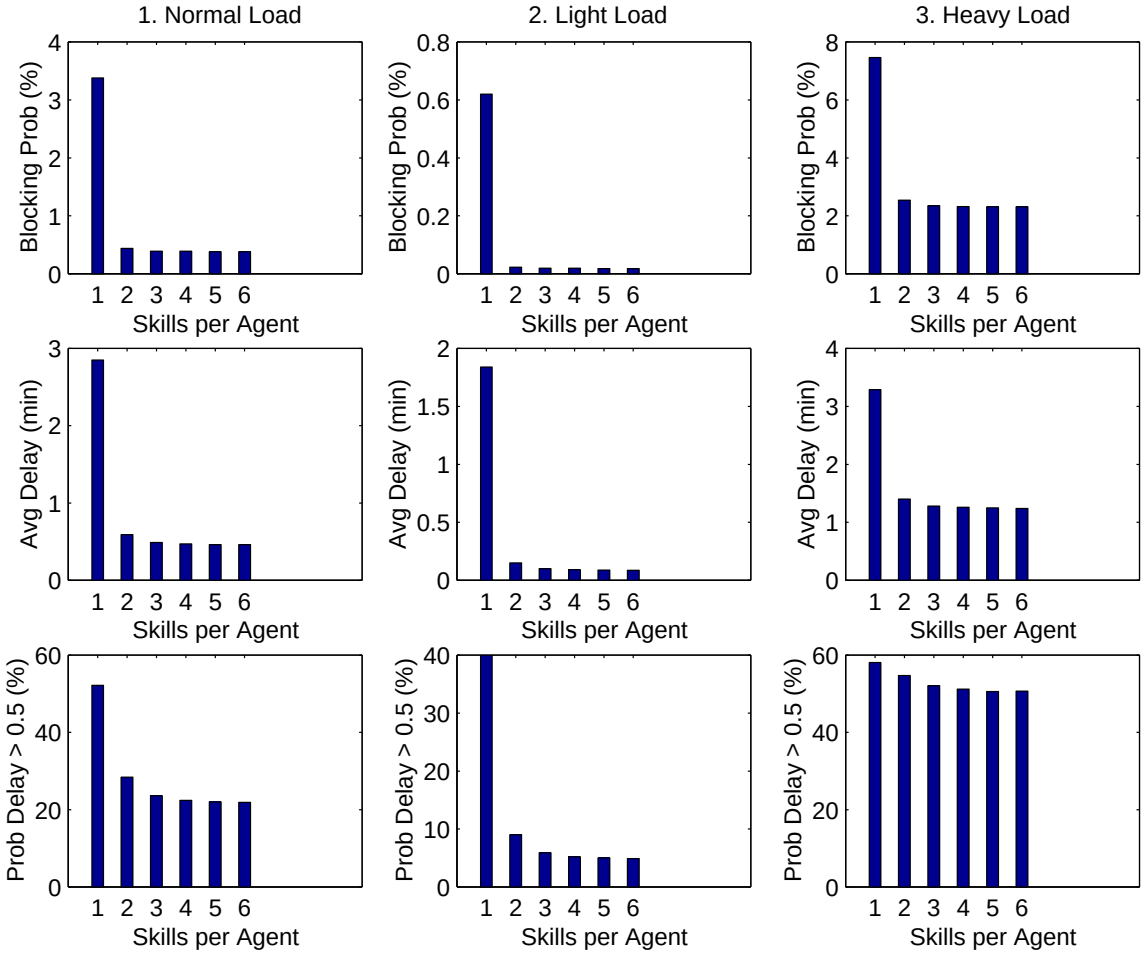

Figure 1: Typical performance measures as a function of the number of skills per agent and the offered load for an $M_6/M/90/30$ SBR call center with $1/\mu_1 = \cdots = 1/\mu_6 = 10$ minutes. The specific performance measures are the blocking probability, the mean conditional delay given entry, and the conditional probability that the delay is greater than 0.5, given entry.

Performance	**Number of Skills per Agent**					
Measure	**One**	**Two**	**Three**	**Four**	**Five**	**Six**
1. Blocking (%)	3.38	0.44	0.39	0.39	0.38	0.38
2. Mean Delay (min)	2.85	0.59	0.49	0.47	0.46	0.46
3. Mean Delay 1	2.99	0.60	0.48	0.46	0.45	0.46
3. Mean Delay 2	2.83	0.60	0.49	0.47	0.46	0.46
3. Mean Delay 3	2.77	0.57	0.47	0.45	0.45	0.44
3. Mean Delay 4	2.78	0.57	0.49	047	0.45	0.45
3. Mean Delay 5	2.92	0.60	0.49	0.45	0.45	0.44
3. Mean Delay 6	2.90	0.61	0.51	0.48	0.47	0.47
4. $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\middle|\, \text{entry}\right)$ (%)	47.8	71.6	76.4	77.6	78.0	78.1
5. $\mathcal{P}\left(\text{Delay}_1 \leq 0.5 \,\middle|\, \text{entry}\right)$	47.3	71.1	76.4	77.5	78.0	77.9
5. $\mathcal{P}\left(\text{Delay}_2 \leq 0.5 \,\middle|\, \text{entry}\right)$	47.2	71.5	76.2	77.6	77.9	78.0
5. $\mathcal{P}\left(\text{Delay}_3 \leq 0.5 \,\middle|\, \text{entry}\right)$	48.3	72.1	76.7	77.9	78.2	78.4
5. $\mathcal{P}\left(\text{Delay}_4 \leq 0.5 \,\middle|\, \text{entry}\right)$	48.7	72.2	76.6	77.6	78.1	78.1
5. $\mathcal{P}\left(\text{Delay}_5 \leq 0.5 \,\middle|\, \text{entry}\right)$	48.3	71.6	76.4	78.1	78.1	78.3
5. $\mathcal{P}\left(\text{Delay}_6 \leq 0.5 \,\middle|\, \text{entry}\right)$	48.0	71.4	76.2	77.5	77.8	78.0
6. Avg Agent Util (%)	89.7	92.8	92.9	92.9	92.9	92.9
7. Work Group 1 Util	89.8	92.9	92.9	92.9	92.9	92.9
7. Work Group 2 Util	89.8	92.8	92.8	92.9	92.9	92.9
7. Work Group 3 Util	89.8	92.8	93.0	92.9	92.9	92.9
7. Work Group 4 Util	89.8	92.9	92.9	92.9	92.9	92.9
7. Work Group 5 Util	89.5	92.8	92.8	92.9	92.9	92.9
7. Work Group 6 Util	89.8	92.9	92.9	92.9	92.9	92.9
8. Work Group 1 Prim Util	89.8	69.4	64.8	63.2	62.4	62.0
8. Work Group 2 Prim Util	89.8	69.2	64.2	63.0	62.0	62.5
8. Work Group 3 Prim Util	89.8	69.2	64.2	62.8	62.2	62.1
8. Work Group 4 Prim Util	89.8	68.8	64.1	63.0	62.1	61.9
8. Work Group 5 Prim Util	89.5	69.1	64.3	62.7	62.2	61.9
8. Work Group 6 Prim Util	89.8	69.5	64.9	62.8	62.4	62.0

Table 1: Performance Measures for the $M_6/M/90/30$ SBR simulation run with offered load $\alpha = 84.0$, mean service times $1/\mu_i = 10$ min, $C_i = 15$ agents ($i = 1, \ldots, 6$). This is the normal-load example.

Performance	**Number of Skills per Agent**					
Measure	**One**	**Two**	**Three**	**Four**	**Five**	**Six**
1. Blocking (%)	0.62	0.023	0.019	0.019	0.018	0.018
2. Mean Delay (min)	1.84	0.15	0.10	0.09	0.09	0.09
3. Mean Delay 1	1.83	0.16	0.10	0.09	0.09	0.09
3. Mean Delay 2	1.70	0.16	0.10	0.09	0.09	0.09
3. Mean Delay 3	1.75	0.15	0.10	0.09	0.09	0.09
3. Mean Delay 4	1.75	0.15	0.10	0.09	0.09	0.09
3. Mean Delay 5	1.99	0.15	0.10	0.09	0.08	0.09
3. Mean Delay 6	2.04	0.15	0.10	0.09	0.09	0.08
4. $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\middle|\, \text{entry}\right)$ (%)	60.0	91.0	94.1	94.8	95.0	95.1
5. $\mathcal{P}\left(\text{Delay}_1 \leq 0.5 \,\middle|\, \text{entry}\right)$	60.3	90.7	94.0	94.7	94.9	94.9
5. $\mathcal{P}\left(\text{Delay}_2 \leq 0.5 \,\middle|\, \text{entry}\right)$	60.8	91.0	94.0	94.8	95.0	95.0
5. $\mathcal{P}\left(\text{Delay}_3 \leq 0.5 \,\middle|\, \text{entry}\right)$	60.6	91.0	94.1	94.9	95.1	95.1
5. $\mathcal{P}\left(\text{Delay}_4 \leq 0.5 \,\middle|\, \text{entry}\right)$	60.7	91.1	94.0	94.8	95.0	95.0
5. $\mathcal{P}\left(\text{Delay}_5 \leq 0.5 \,\middle|\, \text{entry}\right)$	59.6	91.1	94.4	94.9	95.3	95.3
5. $\mathcal{P}\left(\text{Delay}_6 \leq 0.5 \,\middle|\, \text{entry}\right)$	59.1	91.1	94.1	94.8	95.0	95.2
6. Avg Agent Util (%)	85.4	86.0	86.0	86.0	86.0	86.0
7. Work Group 1 Util	85.3	86.2	86.0	85.9	85.9	86.0
7. Work Group 2 Util	85.2	85.9	86.0	85.9	85.9	86.0
7. Work Group 3 Util	85.3	85.9	85.9	85.8	85.9	85.9
7. Work Group 4 Util	85.2	85.9	86.0	86.1	86.0	86.0
7. Work Group 5 Util	85.5	86.0	85.9	86.0	86.0	85.9
7. Work Group 6 Util	85.6	85.9	85.9	86.1	86.0	85.9
8. Work Group 1 Prim Util	85.3	69.3	66.5	65.6	65.3	65.1
8. Work Group 2 Prim Util	85.2	69.1	66.4	65.6	65.2	64.9
8. Work Group 3 Prim Util	85.3	68.9	66.2	65.3	65.0	65.0
8. Work Group 4 Prim Util	85.2	69.0	66.3	65.7	65.3	65.3
8. Work Group 5 Prim Util	85.5	69.0	66.4	65.6	65.3	65.1
8. Work Group 6 Prim Util	85.6	68.9	66.4	65.7	65.4	64.9

Table 2: Performance Measures for the $M_6/M/90/30$ SBR simulation run with offered load $\alpha = 77.4$, mean service times $1/\mu_i = 10$ min, $C_i = 15$ agents ($i = 1, \ldots, 6$). This is the light-load example.

Performance	**Number of Skills per Agent**					
Measure	**One**	**Two**	**Three**	**Four**	**Five**	**Six**
1. Blocking (%)	7.46	2.54	2.35	2.32	2.31	2.31
2. Mean Delay (min)	3.29	1.40	1.28	1.26	1.25	1.24
3. Mean Delay 1	3.36	1.42	1.31	1.31	1.27	1.31
3. Mean Delay 2	3.30	1.37	1.27	1.26	1.24	1.25
3. Mean Delay 3	3.06	1.41	1.27	1.22	1.25	1.20
3. Mean Delay 4	3.28	1.35	1.28	1.23	1.23	1.22
3. Mean Delay 5	3.29	1.39	1.27	1.28	1.23	1.25
3. Mean Delay 6	3.59	1.43	1.29	1.24	1.26	1.24
4. $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\middle|\, \text{entry}\right)$ (%)	41.9	45.3	47.9	48.8	49.4	49.3
5. $\mathcal{P}\left(\text{Delay}_1 \leq 0.5 \,\middle|\, \text{entry}\right)$	41.1	45.1	47.7	48.4	49.1	49.3
5. $\mathcal{P}\left(\text{Delay}_2 \leq 0.5 \,\middle|\, \text{entry}\right)$	42.3	45.0	48.1	48.7	49.1	49.0
5. $\mathcal{P}\left(\text{Delay}_3 \leq 0.5 \,\middle|\, \text{entry}\right)$	42.7	45.2	48.1	49.3	49.5	50.0
5. $\mathcal{P}\left(\text{Delay}_4 \leq 0.5 \,\middle|\, \text{entry}\right)$	42.1	45.7	48.0	49.2	49.8	49.6
5. $\mathcal{P}\left(\text{Delay}_5 \leq 0.5 \,\middle|\, \text{entry}\right)$	41.8	45.4	47.9	48.8	49.5	48.9
5. $\mathcal{P}\left(\text{Delay}_6 \leq 0.5 \,\middle|\, \text{entry}\right)$	41.3	45.4	48.2	48.8	49.5	49.4
6. Avg Agent Util (%)	91.8	97.4	97.6	97.6	97.6	97.6
7. Work Group 1 Util	92.1	97.4	97.6	97.6	97.6	97.6
7. Work Group 2 Util	91.6	97.4	97.6	97.6	97.6	97.6
7. Work Group 3 Util	91.5	97.4	97.6	97.6	97.6	97.6
7. Work Group 4 Util	91.9	97.4	97.6	97.6	97.7	97.7
7. Work Group 5 Util	91.8	97.4	97.6	97.6	97.7	97.7
7. Work Group 6 Util	91.8	97.4	97.6	97.6	97.6	97.7
8. Work Group 1 Prim Util	92.1	73.8	69.3	68.9	67.7	67.6
8. Work Group 2 Prim Util	91.6	73.6	69.4	68.3	67.6	67.4
8. Work Group 3 Prim Util	91.5	73.3	69.1	67.9	67.5	67.1
8. Work Group 4 Prim Util	91.9	73.4	69.3	68.2	67.3	67.6
8. Work Group 5 Prim Util	91.8	73.7	69.2	68.4	67.6	67.9
8. Work Group 6 Prim Util	91.8	73.7	70.0	67.9	67.4	67.6

Table 3: Performance Measures for the $M_6/M/90/30$ SBR simulation run with offered load $\alpha = 90.0$, mean service times $1/\mu_i = 10$ min, $C_i = 15$ agents ($i = 1, \ldots, 6$). This is the heavy-load example.

		Performance Measure			
Loading	Model	Blocking	$E[W]$	$P(W \leq 0.5)$	Utiliz.
	$M/M/90/30$	0.00017	0.08	0.942	0.860
	6 skills	0.00018	0.09	0.951	0.860
	2 skills	0.00023	0.15	0.910	0.860
light	1 skill	0.0062	1.84	0.600	0.854
	$M/M/15/5$	0.0388	0.59	0.737	0.854
	$M/M/15/30$	0.00007	2.15	0.575	0.827
	$M/M/90/30$	0.0036	0.45	0.733	0.930
	6 skills	0.0038	0.46	0.781	0.929
	2 skills	0.0044	0.59	0.716	0.928
normal	1 skill	0.0336	2.85	0.478	0.897
	$M/M/15/5$	0.065	0.82	0.641	0.927
	$M/M/15/30$	0.0066	4.94	0.344	0.872
	$M/M/90/30$	0.024	1.24	0.387	0.977
	6 skills	0.023	1.24	0.493	0.976
	2 skills	0.025	1.40	0.453	0.974
heavy	1 skill	0.075	3.29	0.419	0.918
	$M/M/15/5$	0.095	1.05	0.555	0.972
	$M/M/15/30$	0.028	8.97	0.153	0.905

Table 4: A comparison of simulation estimates of performance measures for the SBR model with 6 skills, 2 skills and 1 skill for three different loadings: light, normal and heavy. Also included are the corresponding exact numerical results for the $M/M/90/30$, $M/M/15/5$ and $M/M/15/30$ models. The waiting time is conditional upon entry.

	M/M/90/30		
Perf. Measure	$\lambda = 7.74$	$\lambda = 8.40$	$\lambda = 9.00$
1. Blocking (%)	0.0168	0.36	2.35
2. Mean Delay (min)	0.083	0.450	1.24
4. $\mathcal{P}\,(\text{Delay} \leq 0.5\,\vert\text{entry})$ (%)	94.2	73.3	38.7
4. $\mathcal{P}\,(\text{Delay} \leq 1.0\,\vert\text{entry})$ (%)	97.0	81.6	49.5
6. Avg. Agent Util. (%)	85.99	93.00	97.65

Table 5: Performance Measures for the $M/M/90/30$ model to put the SBR simulation results in perspective. The mean service time is $1/\mu = 10$ minutes.

	M/M/15/K					
	K=5			K=30		
Perf. Measure	$\lambda = 1.29$	$\lambda = 1.40$	$\lambda = 1.50$	$\lambda = 1.29$	$\lambda = 1.40$	$\lambda = 1.50$
1. Blocking (%)	3.88	6.53	9.48	0.0073	0.66	2.81
2. Mean Delay (min)	0.585	0.823	1.05	2.15	4.94	8.97
4. $\mathcal{P}\,(\text{Delay} \leq 0.5\,\vert\text{entry})$ (%)	73.7	64.1	55.5	57.5	34.4	15.3
4. $\mathcal{P}\,(\text{Delay} \leq 1.0\,\vert\text{entry})$ (%)	79.1	70.9	63.3	61.8	38.1	17.5
6. Avg. Agent Util. (%)	85.37	92.72	97.19	82.66	87.24	90.52

Table 6: Performance Measures for the $M/M/15/K$ model for $K = 5$ and $K = 30$ to put the SBR simulation results for the case in which all agents having only one skill ($j = 1$) in perspective. The mean service time is $1/\mu = 10$ minutes.

both the performance in the SBR model with 6 skills and the performance in the $M/M/90/30$ model. On the other hand, the performance in these cases is not close to the performance in the other three cases, and the performance in those three cases varies widely.

We present a few additional details about these $M/M/C/K$ models in Tables 5 and 6.

3. A Balanced Example for the Provisioning Algorithm

In this section we show how the provisioning algorithm in Section 5 of the main paper works in a balanced example, closely related to the example used for the resource-pooling experiment in Section 4 of the main paper. As before, the balanced example has 6 call types. Again the mean service time is 10 minutes. Here the offered load for each call type is 13.75, so that the total offered load is 82.5.

We consider the two performance constraints in (2.1) and (2.2) in the main paper, which we now review. The specific performance constraints we consider are *speed-to-answer service-*

level constraints and *blocking-probability constraints*. To state these constraints, let W_k be the steady-state waiting time before beginning service and let Q_k be the steady-state number of customers in the system, experienced by an arrival of type k. The per-class speed-to-answer service-level constraints specify that the *conditional* probability that the steady-state waiting time for type-k calls is below a desired target, given that the call is not blocked, should be above some threshold, i.e.,

$$P(W_k \le \tau_k | Q_k < C + K) \ge \delta_k, \quad 1 \le k \le n \ , \tag{3.1}$$

where τ_k is the type-k target and δ_k is the type-k threshold. The per-class blocking probability constraints are

$$P(Q_k = C + K) \le \epsilon_k, \quad 1 \le k \le n \ . \tag{3.2}$$

Here we let $\tau_k = \tau = 0.5$ and $\delta_k = \delta = 0.80$ for all k, which corresponds to the requirement that 80% of the calls of each type be answered within 0.5 minute (30 seconds).

Under the assumption of Poisson arrivals (which we have assumed here, but which need not hold in practice), the random variables Q_k are distributed as Q, the steady-state total number in the system at an arbitrary time, by virtue of the *Poisson Arrivals See Time Averages (PASTA)* property; e.g., see Wolff (1989). Here we will let the blocking probability target be $\epsilon_k = \epsilon = 0.005$ for all k, which corresponds to 0.5% blocking. Typically, agents are much more expensive than trunk lines, so that the blocking probability target should be relatively small. The blocking-probability constraint is included so that there are not substantially more trunk lines than needed.

We start by applying the $M/M/C/K$ model. First, the asymptotic method for the $M/M/C/K$ model in Massey and Wallace (2004) yields an initial solution of $(C, K) = (90, 21)$. As shown in Table 7 exact analysis yields $(90, 20)$ instead, but the blocking probability is close to the boundary (0.5% blocking). Clearly, 89 agents are insufficient; the delay constraint and the blocking constraint are simultaneously violated.

In contrast, if all agents have only one skill, then we have 6 separate $M/M/C/K$ models each with offered load 13.75, for which the optimal solution is $(18, 10)$, yielding a total $(C, K) = (108, 60)$. Of course, since we have a common waiting room, we would not need $K = 60$. The main point is that we would need 18 more agents if each agent had only one skill. So, clearly, adding multiple skills has an enormous performance impact. Indeed, 20% more agents are required when agents have only one skill instead of all six skills. The question is whether we can experience much less performance degradation when agents have only two skills.

	M/M/C/K and $\lambda = 8.25$			
Perf. Measure	$C = 90, K = 21$	$C = 90, K = 20$	$C = 90, K = 19$	$C = 89, K = 21$
1. Blocking (%)	0.45	0.49	0.53	0.60
2. Mean Delay (min)	0.248	0.238	0.227	0.303
4. $\mathcal{P}\,(\text{Delay} \leq 0.5\,\vert\text{entry}\,)$ (%)	82.4	82.9	83.2	78.9
4. $\mathcal{P}\,(\text{Delay} \leq 1.0\,\vert\text{entry}\,)$ (%)	89.6	90.0	90.5	87.0
6. Avg. Agent Util. (%)	91.25	91.22	91.18	91.12

Table 7: Performance Measures for the $M/M/C/K$ model for different C and K to put the SBR simulation results in perspective. The mean service time is $1/\mu = 10$ minutes.

We now turn to the simulation phase of the algorithm. For the initial value $C = 90$, we can assign primary and secondary skills to each agent in a balanced way. Each of the 6 work groups has 15 agents. And, for each work group (primary skill), each of the 5 remaining skills are assigned to 3 agents in the work group. Thus each of the 30 pairs of distinct skills are assigned as primary and secondary skills to 3 agents. Given that obvious initial agent skill matrix, we apply simulation to evaluate its performance. The simulation results for this initial case with $(C, K) = (90, 21)$ and that skill matrix is given in Table 8. We show the overall blocking probability, the overall mean delay, the mean delay for each call type, the overall speed-to-answer service-level $\mathcal{P}\,(\text{Delay} \leq 0.5\,\vert\text{entry}\,)$, the per-call-type speed-to-answer service-level $\mathcal{P}\,(\text{Delay}_i \leq 0.5\,\vert\text{entry}\,)$, the overall agent utilization percentage, the work-group utilization percentages and the percentage work-group utilizations devoted to call types with that skill as a primary skill. We do not display confidence intervals, but since the model is symmetric, we can see the statistical precision form the six per-class results.

From Table 8, we see that the blocking probability is 0.0054, which is above the target $\epsilon = 0.0050$, while some of the speed-to-answer service-level probabilities are also below the target 80%. However, the constraints are only violated by very small amounts. Indeed, those speed-to-answer discrepancies could conceivably be due to statistical error, but the blocking gap seems to be statistically (if not practically) significant. For all practical purposes the initial candidate solution based on the $M/M/C/K$ model does the job for this balanced example.

However, we will proceed until all constraints are fully satisfied. Following the specified procedure, we increment C by 1 and decrement K by one, to produce the new candidate pair $(91, 20)$. It next remains to specify the work groups and the agent-skill matrix. In the main paper we used a performance-based method. However, here we describe an alternative fair-allocation method, which was initially our preferred algorithm. The tables here were

developed based on that initial alternative algorithm. In subsequent experiments we found that the performance-based method tended to be more efficient, leading to fewer iterations overall.

So we start by specifying the alternative fair-allocation method. The fair-allocation method proceeds to allocate primary and secondary skills in precisely the same way we did for the initial candidate solution. All that changes is the pair (C, K). Here, instead of $(90, 21)$, we have $(91, 20)$. For this symmetric example, we use the fair-allocation method and get $R_i = 15.1667$ for all i. We thus, arbitrarily add one agent to work group 6. We then arbitrarily give this agent secondary skill 5, so we add the row $(6, 5, 0, 0, 0, 0)$ to the initial agent-skill matrix. Then we simulate this new case and obtain the results in Table 8. And now we see that the performance constraints are all satisfied, so we can stop. As we should expect, the performance is somewhat better for work groups 6 and 5 because they received the extra help. The main point, though, is that the provisioning solution when each agent has only two skills is nearly the same as for the single-class $M/M/C/K$ model. *There is only a difference of a single agent!*

To further investigate what is going on, we also apply our staffing algorithm to the case in which all agents are universal agents, i.e., in which all agents have 6 skills. Interestingly, we find that the best feasible solution has a strictly smaller value of C than for the $M/M/C/K$ model. Our provisioning algorithm terminates with the pair $(C, K) = (89, 28)$. It turns out that the non-FCFS service discipline of the 6-skill policy requires one fewer agent (but with extra trunk lines). We performed additional simulation experiments to verify that this is a bonafide phenomenon of the routing policy. Even so, the staffing with two skills differs by only 2 agents from the staffing with six skills. Moreover, the application of the $M/M/C/K$ model to find an initial candidate solution gets very close to the final answer. Only one more iteration was required to reach the best feasible solution.

4. An Unbalanced Example

In this section we consider a more difficult unbalanced example. We leave the mean service times the same, but modify the arrival rates so that the offered loads become

$$\alpha_1 = \alpha_2 = 4.25, \quad \alpha_3 = 10.50, \quad \alpha_4 = 13.75, \quad \alpha_5 = 19.25 \quad \text{and} \quad \alpha_6 = 30.50 \tag{4.1}$$

Just as for the balanced example, the total offered load is 82.50, but now the six offered loads are unbalanced.

The SBR Provisioning Algorithm for the Balanced Example			
	Experiment		
Performance Measure	**iteration 1 C = 90, K = 21**	**iteration 2 C = 91, K = 20**	**6 skills C=89, K = 28**
1. Blocking (%)	0.54	0.43	0.36
2. Mean Delay (min)	0.36	0.30	0.39
3. Mean Delay 1	0.37	0.32	0.38
3. Mean Delay 2	0.36	0.32	0.37
3. Mean Delay 3	0.35	0.30	0.40
3. Mean Delay 4	0.36	0.30	0.38
3. Mean Delay 5	0.35	0.28	0.39
3. Mean Delay 6	0.37	0.28	0.39
4. $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\middle|\text{entry}\right)$ (%)	79.8	82.7	80.8
5. $\mathcal{P}\left(\text{Delay}_1 \leq 0.5 \,\middle|\text{entry}\right)$	79.5	81.9	80.8
5. $\mathcal{P}\left(\text{Delay}_2 \leq 0.5 \,\middle|\text{entry}\right)$	79.7	81.9	81.2
5. $\mathcal{P}\left(\text{Delay}_3 \leq 0.5 \,\middle|\text{entry}\right)$	80.0	82.5	80.5
5. $\mathcal{P}\left(\text{Delay}_4 \leq 0.5 \,\middle|\text{entry}\right)$	80.0	82.6	81.0
5. $\mathcal{P}\left(\text{Delay}_5 \leq 0.5 \,\middle|\text{entry}\right)$	80.3	83.5	80.8
5. $\mathcal{P}\left(\text{Delay}_6 \leq 0.5 \,\middle|\text{entry}\right)$	79.7	83.9	80.5
6. Avg Agent Util (%)	91.1	90.2	92.1
7. Work Group 1 Util	91.2	90.4	91.9
7. Work Group 2 Util	91.2	90.3	92.1
7. Work Group 3 Util	91.1	90.4	92.4
7. Work Group 4 Util	91.2	90.3	92.0
7. Work Group 5 Util	91.2	90.4	92.0
7. Work Group 6 Util	91.2	89.7	92.0
8. Work Group 1 Prim Util	68.3	69.1	61.8
8. Work Group 2 Prim Util	68.0	68.9	62.0
8. Work Group 3 Prim Util	67.8	68.7	63.7
8. Work Group 4 Prim Util	67.8	68.9	61.7
8. Work Group 5 Prim Util	67.9	68.3	62.2
8. Work Group 6 Prim Util	68.3	66.5	61.7

Table 8: Performance measures for the balanced-offered-load example in which the offered load are $\alpha_1 = \cdots = \alpha_6 = 13.75$, the mean service times are $1/\mu_1 = \cdots = 1/\mu_6 = 10.0$ min, the blocking-probability target is $\epsilon = 0.005$ and the speed-to-answer service-level target is $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\middle|\text{entry}\right) \geq 0.80$. The last column gives the six-skill best feasible solution for comparison.

		Square-Root Method									
		$C = 90$		$C = 91$		$C = 92$		$C = 93$		$C = 94$	
		$x = 0.358$		$x = 0.405$		$x = 0.453$		$x = 0.500$		$x = 0.548$	
No.	α_i	**Real**	C_i	**Real**	C_i	**Real**	C_i	**Real**	C_i	**Real**	C_i
1.	4.25	4.99	5	5.09	6	5.18	6	5.28	6	5.38	6
2.	4.25	4.99	5	5.09	6	5.18	6	5.28	6	5.38	6
3.	10.50	11.66	12	11.81	12	11.97	12	12.12	13	12.28	13
4.	13.75	15.08	16	15.25	16	15.43	16	15.61	16	15.78	16
5.	19.25	20.82	21	21.03	22	21.24	22	21.45	22	21.66	22
6.	30.50	32.47	31	32.74	29	33.00	30	33.26	30	33.53	31
total	82.5	90.00	90	91.00	91	92.00	92	93.00	93	94.00	94

Table 9: The number of agents in each work group computed by the square-root method for the unbalanced-offered-load example, where the rounding is done by rounding up the first five work groups and then compensating by rounding down the sixth work group.

Just as for the balanced example, if all agents have all six skills, the solution based on the $M/M/C/K$ model is $(C, K) = (90, 21)$. On the other hand, if each agent has only a single skill, then we find the six required (C, K) pairs associated with the offered loads in (4.1) are: $(7, 5)$, $(7, 5)$, $(14, 8)$, $(18, 9)$, $(24, 10)$, and $(36, 13)$, respectively, yielding a total requirement of $(106, 50)$. Interestingly, fewer agents are required in the single-skill case for the unbalanced model than for the balanced model (106 instead of 108), but there still is a dramatic increase in required resources.

To see what happens when each agent has two skills, we again turn to the simulation. However, in the unbalanced case the initial case is no longer symmetric, so from the outset we have rounding problems when we specify the work groups. The second step to find an initial feasible solution phase, using the fair-allocation method, takes four iterations, proceeding from $C = 90$ to $C = 93$.

Given the work-group sizes specified in the $C = 90$ column of Table 9, we next define the initial agent-skill matrix A. Following either equation (6.4) or (6.5) in the main paper, we assign secondary skills to the agents as specified in Table 10. The initial agent-skill matrix itself is displayed in Appendix B.

As indicated, the initial phase of the simulation algorithm in which we simply increment C takes us to the initial feasible solution $(93, 18)$ in four steps. (As for the balanced example, we use the fair-allocation method here.) The second agent-skill matrix is also shown in Appendix B; it has three new rows, which are highlighted in boldface. The next two cases for $(92, 19)$ and $(93, 18)$ each involve the addition of a single row. The added rows in these two steps are,

Primary	**Number of secondary skills that support call-type k**					
Skills	$k = \mathbf{1}$	$k = \mathbf{2}$	$k = \mathbf{3}$	$k = \mathbf{4}$	$k = \mathbf{5}$	$k = \mathbf{6}$
$C_1 = 5$	$C_{1,2,1} = 0$	$C_{1,2,2} = 1$	$C_{1,2,3} = 1$	$C_{1,2,4} = 1$	$C_{1,2,5} = 1$	$C_{1,2,6} = 1$
$C_2 = 5$	$C_{2,2,1} = 1$	$C_{2,2,2} = 0$	$C_{2,2,3} = 1$	$C_{2,2,4} = 1$	$C_{2,2,5} = 1$	$C_{2,2,6} = 1$
$C_3 = 12$	$C_{321} = 1$	$C_{322} = 1$	$C_{323} = 0$	$C_{324} = 2$	$C_{325} = 3$	$C_{326} = 5$
$C_4 = 16$	$C_{421} = 1$	$C_{422} = 1$	$C_{423} = 3$	$C_{424} = 0$	$C_{425} = 4$	$C_{426} = 7$
$C_5 = 21$	$C_{521} = 2$	$C_{522} = 1$	$C_{523} = 4$	$C_{524} = 5$	$C_{525} = 0$	$C_{526} = 9$
$C_6 = 31$	$C_{621} = 3$	$C_{622} = 3$	$C_{623} = 6$	$C_{624} = 8$	$C_{625} = 11$	$C_{626} = 0$
total $C = 90$	8	7	15	17	20	23

Table 10: Specification of the secondary skills for the initial agent-skill matrix $A^{(2)}_{90\times 6}$ using the recommended method for evenly assigning agents skills, for the unbalanced example.

respectively, $(6, 4, 0, 0, 0, 0)$ and $(3, 4, 0, 0, 0, 0)$. Those additions to the agent-skill matrix for $(91, 20)$ give us the skill matrix for $(93, 18)$.

The performance measures for each case are shown in Table 11. From Table 11, we see that the last $(93, 18)$ case is indeed feasible, but some of the speed-to-answer service-level probabilities are much above target, suggesting that we might well be able to reduce the number of agents.

We now turn to the second phase of the resource-provisioning algorithm. Now we make local adjustments based on the observed performance, i.e., we find better feasible solutions. We start with the $(93, 18)$ feasible solution that concluded the incrementing-C phase. The successive changes we make in this second performance-based phase are displayed in Table 12. In the 8^{th} iteration (simulation run) we obtain a feasible solution with the pair $(91, 20)$, once again coming within a single agent of the single-group $M/M/C/K$ optimal feasible solution $(90, 21)$. The final feasible agent-skill matrix is displayed in Appendix B.

To verify the we have found the best possible solution (within our local-search framework), we continue to perform simulations. We go on to a 9^{th} iteration, deleting the row $(3, 5, 0, 0, 0, 0)$, but we see that there is no point to try to achieve $C = 90$, because the blocking probability exceeds the target 0.5% and the overall speed-to-answer service-level probability is under target, as well as several of the individual call-type speed-to-answer service-level probabilities. And there is no longer any significant slack in the individual call-type speed-to-answer service-level probabilities. The highest one is 80.5%. Indeed, further change steps do not yield a feasible solution.

For this unbalanced example we also applied the algorithm to the 6-skill case, in which all

Finding an Initial Feasible Solution				
	Number of Iterations (No. of Agents)			
Performance Measure	**1 (90)**	**2 (91)**	**3 (92)**	**4 (93)**
1. Blocking (%)	0.53	0.42	0.36	0.30
2. Mean Delay (min)	0.35	0.28	0.23	0.18
3. Mean Delay 1	0.82	0.55	0.47	0.40
3. Mean Delay 2	0.97	0.58	0.48	0.40
3. Mean Delay 3	0.39	0.33	0.27	0.20
3. Mean Delay 4	0.31	0.28	0.22	0.18
3. Mean Delay 5	0.27	0.22	0.19	0.16
3. Mean Delay 6	0.24	0.22	0.17	0.13
4. $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\middle|\text{entry}\right)$ (%)	81.3	83.9	86.5	88.8
5. $\mathcal{P}\left(\text{Delay}_1 \leq 0.5 \,\middle|\text{entry}\right)$	68.3	75.5	78.4	80.5
5. $\mathcal{P}\left(\text{Delay}_2 \leq 0.5 \,\middle|\text{entry}\right)$	65.2	74.9	77.8	80.3
5. $\mathcal{P}\left(\text{Delay}_3 \leq 0.5 \,\middle|\text{entry}\right)$	79.7	81.8	84.7	88.0
5. $\mathcal{P}\left(\text{Delay}_4 \leq 0.5 \,\middle|\text{entry}\right)$	82.0	83.6	86.5	88.8
5. $\mathcal{P}\left(\text{Delay}_5 \leq 0.5 \,\middle|\text{entry}\right)$	83.4	86.2	87.8	89.8
5. $\mathcal{P}\left(\text{Delay}_6 \leq 0.5 \,\middle|\text{entry}\right)$	84.4	85.8	88.7	90.9
6. Avg Agent Util (%)	91.2	90.2	89.3	88.4
7. Work Group 1 Util	86.8	84.9	83.2	82.1
7. Work Group 2 Util	86.8	84.6	83.0	81.8
7. Work Group 3 Util	89.5	88.6	87.8	86.0
7. Work Group 4 Util	90.2	89.5	88.6	87.7
7. Work Group 5 Util	91.6	90.5	89.6	88.8
7. Work Group 6 Util	93.4	93.5	92.6	92.1
8. Work Group 1 Prim Util	54.4	50.4	50.3	51.6
8. Work Group 2 Prim Util	55.5	50.3	51.1	50.8
8. Work Group 3 Prim Util	63.7	63.6	64.6	62.7
8. Work Group 4 Prim Util	66.4	67.1	67.6	67.9
8. Work Group 5 Prim Util	72.0	70.6	71.6	72.0
8. Work Group 6 Prim Util	78.8	80.8	80.5	81.1

Table 11: Performance measures in the initial phase of the algorithm for the unbalanced example having offered loads $\alpha_1 = \alpha_2 = 4.25$, $\alpha_3 = 10.50$, $\alpha_4 = 13.75$, $\alpha_5 = 19.25$, and $\alpha_6 = 30.50$. As always, the mean service times are $1/\mu_1 = \cdots = 1/\mu_6 = 10.0$ minutes, the blocking-probability target is $\epsilon = 0.005$ and the speed-to-answer service-level target is target delay $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\middle|\text{entry}\right) \geq 0.80$.

Steps in the Second Refinement Phase of the Algorithm														
Iteration	**Agent Removal**	**Skill Change**	$\mathbf{a}^T =$						$\mathbf{b}^T =$						(C, K)
			j	k	0	0	0	0	q	r	0	0	0	0	
5	√		6	5	0	0	0	0							(92, 19)
6		√	6	5	0	0	0	0	2	1	0	0	0	0	(92, 19)
7	√		4	6	0	0	0	0							(91, 20)
8		√	5	6	0	0	0	0	1	2	0	0	0	0	(91, 20)
9	√		3	5	0	0	0	0							(90, 21)

Table 12: The agent-skill-matrix updates during the second performance-based refinement phase of the algorithm. The row vectors $\mathbf{a}^T$ and $\mathbf{b}^T$ are the deleted and inserted rows, respectively, corresponding to the skill profiles of agents in the highest and worst performing work groups.

agents are universal agents. Just as for the balanced example, the best feasible solution had $C = 89$, one fewer agent than in the single-group $M/M/C/K$ case. In this case, $K = 24$. Again we see that the SBR provisioning algorithm is remarkably effective.

Finding Better Feasible Solutions						
	Number of Iterations (No of Agents)					
Performance Measure	**4 (93)**	**5 (92)**	**6 (92)**	**7 (91)**	**8 (91)**	**9 (90)**
1. Blocking (%)	0.30	0.35	0.36	0.43	0.44	0.54
2. Mean Delay (min)	0.18	0.23	0.23	0.28	0.29	0.36
3. Mean Delay 1	0.40	0.47	0.38	0.47	0.37	0.40
3. Mean Delay 2	0.40	0.48	0.37	0.46	0.36	0.41
3. Mean Delay 3	0.20	0.23	0.24	0.29	0.29	0.41
3. Mean Delay 4	0.18	0.21	0.21	0.30	0.30	0.35
3. Mean Delay 5	0.16	0.19	0.21	0.25	0.28	0.35
3. Mean Delay 6	0.13	0.18	0.21	0.25	0.28	0.33
4. $\mathcal{P}\,(\text{Delay} \leq 0.5\,\vert\text{entry}\,)$ (%)	88.8	86.5	86.2	83.4	82.9	79.8
5. $\mathcal{P}\,(\text{Delay}_1 \leq 0.5\,\vert\text{entry}\,)$	80.5	78.0	81.6	78.6	82.6	80.0
5. $\mathcal{P}\,(\text{Delay}_2 \leq 0.5\,\vert\text{entry}\,)$	80.3	77.6	81.4	78.6	81.9	79.7
5. $\mathcal{P}\,(\text{Delay}_3 \leq 0.5\,\vert\text{entry}\,)$	88.0	86.1	85.8	83.6	83.4	78.6
5. $\mathcal{P}\,(\text{Delay}_4 \leq 0.5\,\vert\text{entry}\,)$	88.8	87.2	87.0	83.2	82.6	80.5
5. $\mathcal{P}\,(\text{Delay}_5 \leq 0.5\,\vert\text{entry}\,)$	89.8	87.7	86.7	84.6	83.1	79.4
5. $\mathcal{P}\,(\text{Delay}_6 \leq 0.5\,\vert\text{entry}\,)$	90.9	88.0	86.9	84.1	82.9	80.3
6. Avg Agent Util (%)	88.4	89.3	89.3	90.2	90.2	91.1
7. Work Group 1 Util	82.1	83.2	82.9	84.6	82.5	84.2
7. Work Group 2 Util	81.8	83.2	81.4	83.1	82.6	83.7
7. Work Group 3 Util	86.0	87.0	87.3	88.4	88.5	90.1
7. Work Group 4 Util	87.7	88.8	88.9	90.2	90.5	91.3
7. Work Group 5 Util	88.8	89.7	90.0	90.7	91.2	92.1
7. Work Group 6 Util	92.1	92.9	93.3	93.8	94.0	94.4
8. Work Group 1 Prim Util	51.6	50.8	49.1	48.7	45.5	44.8
8. Work Group 2 Prim Util	50.8	50.5	46.7	46.7	45.6	44.5
8. Work Group 3 Prim Util	62.7	61.9	61.7	61.4	60.9	62.8
8. Work Group 4 Prim Util	67.9	67.0	66.9	68.2	68.2	67.2
8. Work Group 5 Prim Util	72.0	71.4	71.8	71.3	72.8	73.2
8. Work Group 6 Prim Util	81.1	81.5	82.5	82.1	82.4	81.8

Table 13: Performance measures for the unbalanced example in the second refinement phase of the provisioning algorithm. The offered loads are $\alpha_1 = \alpha_2 = 4.25$, $\alpha_3 = 10.50$, $\alpha_4 = 13.75$, $\alpha_5 = 19.25$, and $\alpha_6 = 30.50$.

5. Acknowledgments

This paper is based on the first author's doctoral dissertation at George Washington University, co-advised by Thomas A. Mazzuchi from George Washington University, William A. Massey from Princeton University and the second author, Ward Whitt from Columbia University.

Appendix A. Agent-Skill Matrices for the Resource-Pooling Experiment

In this appendix we display the six agent-skill matrices used in the resource-pooling experiment in Section 2.

$$\mathbf{A}^{(1)}_{\mathbf{90\times 6}} = \begin{pmatrix} \mathbf{B}^{(1)}_{\mathbf{45\times 6}} \\ \mathbf{C}^{(1)}_{\mathbf{45\times 6}} \end{pmatrix}, \mathbf{B}^{(1)}_{\mathbf{45\times 6}} = \begin{pmatrix}
1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\1&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\2&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0\\3&0&0&0&0&0
\end{pmatrix}, \mathbf{C}^{(1)}_{\mathbf{45\times 6}} = \begin{pmatrix}
4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\4&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\5&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0\\6&0&0&0&0&0
\end{pmatrix}.$$

$$\mathbf{A}^{(2)}_{90\times 6} = \begin{pmatrix} \mathbf{B}^{(2)}_{45\times 6} \\ \mathbf{C}^{(2)}_{45\times 6} \end{pmatrix}, \mathbf{B}^{(2)}_{45\times 6} = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 & 0 & 0 \\ 2 & 6 & 0 & 0 & 0 & 0 \\ 2 & 6 & 0 & 0 & 0 & 0 \\ 2 & 6 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 & 0 & 0 \\ 3 & 6 & 0 & 0 & 0 & 0 \\ 3 & 6 & 0 & 0 & 0 & 0 \\ 3 & 6 & 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{C}^{(2)}_{45\times 6} = \begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 & 0 & 0 \\ 4 & 2 & 0 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 \\ 5 & 2 & 0 & 0 & 0 & 0 \\ 5 & 2 & 0 & 0 & 0 & 0 \\ 5 & 2 & 0 & 0 & 0 & 0 \\ 5 & 3 & 0 & 0 & 0 & 0 \\ 5 & 3 & 0 & 0 & 0 & 0 \\ 5 & 3 & 0 & 0 & 0 & 0 \\ 5 & 4 & 0 & 0 & 0 & 0 \\ 5 & 4 & 0 & 0 & 0 & 0 \\ 5 & 4 & 0 & 0 & 0 & 0 \\ 5 & 6 & 0 & 0 & 0 & 0 \\ 5 & 6 & 0 & 0 & 0 & 0 \\ 5 & 6 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 0 & 0 & 0 \\ 6 & 2 & 0 & 0 & 0 & 0 \\ 6 & 2 & 0 & 0 & 0 & 0 \\ 6 & 2 & 0 & 0 & 0 & 0 \\ 6 & 3 & 0 & 0 & 0 & 0 \\ 6 & 3 & 0 & 0 & 0 & 0 \\ 6 & 3 & 0 & 0 & 0 & 0 \\ 6 & 4 & 0 & 0 & 0 & 0 \\ 6 & 4 & 0 & 0 & 0 & 0 \\ 6 & 4 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

$$\mathbf{A}^{(3)}_{90\times 6} = \begin{pmatrix} \mathbf{B}^{(3)}_{45\times 6} \\ \mathbf{C}^{(3)}_{45\times 6} \end{pmatrix}, \mathbf{B}^{(3)}_{45\times 6} = \begin{pmatrix} 1 & 2 & 3 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 \\ 1 & 3 & 4 & 0 & 0 & 0 \\ 1 & 3 & 4 & 0 & 0 & 0 \\ 1 & 3 & 4 & 0 & 0 & 0 \\ 1 & 4 & 5 & 0 & 0 & 0 \\ 1 & 4 & 5 & 0 & 0 & 0 \\ 1 & 4 & 5 & 0 & 0 & 0 \\ 1 & 5 & 6 & 0 & 0 & 0 \\ 1 & 5 & 6 & 0 & 0 & 0 \\ 1 & 5 & 6 & 0 & 0 & 0 \\ 1 & 6 & 2 & 0 & 0 & 0 \\ 1 & 6 & 2 & 0 & 0 & 0 \\ 1 & 6 & 2 & 0 & 0 & 0 \\ 2 & 1 & 3 & 0 & 0 & 0 \\ 2 & 1 & 3 & 0 & 0 & 0 \\ 2 & 1 & 3 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 0 \\ 2 & 5 & 6 & 0 & 0 & 0 \\ 2 & 5 & 6 & 0 & 0 & 0 \\ 2 & 5 & 6 & 0 & 0 & 0 \\ 2 & 6 & 1 & 0 & 0 & 0 \\ 2 & 6 & 1 & 0 & 0 & 0 \\ 2 & 6 & 1 & 0 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 \\ 3 & 1 & 2 & 0 & 0 & 0 \\ 3 & 2 & 4 & 0 & 0 & 0 \\ 3 & 2 & 4 & 0 & 0 & 0 \\ 3 & 2 & 4 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 3 & 5 & 6 & 0 & 0 & 0 \\ 3 & 5 & 6 & 0 & 0 & 0 \\ 3 & 5 & 6 & 0 & 0 & 0 \\ 3 & 6 & 1 & 0 & 0 & 0 \\ 3 & 6 & 1 & 0 & 0 & 0 \\ 3 & 6 & 1 & 0 & 0 & 0 \end{pmatrix}, \mathbf{C}^{(3)}_{45\times 6} = \begin{pmatrix} 4 & 1 & 2 & 0 & 0 & 0 \\ 4 & 1 & 2 & 0 & 0 & 0 \\ 4 & 1 & 2 & 0 & 0 & 0 \\ 4 & 2 & 3 & 0 & 0 & 0 \\ 4 & 2 & 3 & 0 & 0 & 0 \\ 4 & 2 & 3 & 0 & 0 & 0 \\ 4 & 3 & 5 & 0 & 0 & 0 \\ 4 & 3 & 5 & 0 & 0 & 0 \\ 4 & 3 & 5 & 0 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 0 \\ 4 & 6 & 1 & 0 & 0 & 0 \\ 4 & 6 & 1 & 0 & 0 & 0 \\ 4 & 6 & 1 & 0 & 0 & 0 \\ 5 & 1 & 2 & 0 & 0 & 0 \\ 5 & 1 & 2 & 0 & 0 & 0 \\ 5 & 1 & 2 & 0 & 0 & 0 \\ 5 & 2 & 3 & 0 & 0 & 0 \\ 5 & 2 & 3 & 0 & 0 & 0 \\ 5 & 2 & 3 & 0 & 0 & 0 \\ 5 & 3 & 4 & 0 & 0 & 0 \\ 5 & 3 & 4 & 0 & 0 & 0 \\ 5 & 3 & 4 & 0 & 0 & 0 \\ 5 & 4 & 6 & 0 & 0 & 0 \\ 5 & 4 & 6 & 0 & 0 & 0 \\ 5 & 4 & 6 & 0 & 0 & 0 \\ 5 & 6 & 1 & 0 & 0 & 0 \\ 5 & 6 & 1 & 0 & 0 & 0 \\ 5 & 6 & 1 & 0 & 0 & 0 \\ 6 & 1 & 2 & 0 & 0 & 0 \\ 6 & 1 & 2 & 0 & 0 & 0 \\ 6 & 1 & 2 & 0 & 0 & 0 \\ 6 & 2 & 3 & 0 & 0 & 0 \\ 6 & 2 & 3 & 0 & 0 & 0 \\ 6 & 2 & 3 & 0 & 0 & 0 \\ 6 & 3 & 4 & 0 & 0 & 0 \\ 6 & 3 & 4 & 0 & 0 & 0 \\ 6 & 3 & 4 & 0 & 0 & 0 \\ 6 & 4 & 5 & 0 & 0 & 0 \\ 6 & 4 & 5 & 0 & 0 & 0 \\ 6 & 4 & 5 & 0 & 0 & 0 \\ 6 & 5 & 1 & 0 & 0 & 0 \\ 6 & 5 & 1 & 0 & 0 & 0 \\ 6 & 5 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

$$\mathbf{A}^{(4)}_{\mathbf{90\times 6}} = \begin{pmatrix} \mathbf{B}^{(4)}_{\mathbf{45\times 6}} \\ \mathbf{C}^{(4)}_{\mathbf{45\times 6}} \end{pmatrix}, \mathbf{B}^{(4)}_{\mathbf{45\times 6}} = \begin{pmatrix}
1 & 2 & 3 & 4 & 0 & 0 \\ 1 & 2 & 3 & 4 & 0 & 0 \\ 1 & 2 & 3 & 4 & 0 & 0 \\
1 & 3 & 4 & 5 & 0 & 0 \\ 1 & 3 & 4 & 5 & 0 & 0 \\ 1 & 3 & 4 & 5 & 0 & 0 \\
1 & 4 & 5 & 6 & 0 & 0 \\ 1 & 4 & 5 & 6 & 0 & 0 \\ 1 & 4 & 5 & 6 & 0 & 0 \\
1 & 5 & 6 & 2 & 0 & 0 \\ 1 & 5 & 6 & 2 & 0 & 0 \\ 1 & 5 & 6 & 2 & 0 & 0 \\
1 & 6 & 2 & 3 & 0 & 0 \\ 1 & 6 & 2 & 3 & 0 & 0 \\ 1 & 6 & 2 & 3 & 0 & 0 \\
2 & 1 & 3 & 4 & 0 & 0 \\ 2 & 1 & 3 & 4 & 0 & 0 \\ 2 & 1 & 3 & 4 & 0 & 0 \\
2 & 3 & 4 & 5 & 0 & 0 \\ 2 & 3 & 4 & 5 & 0 & 0 \\ 2 & 3 & 4 & 5 & 0 & 0 \\
2 & 4 & 5 & 6 & 0 & 0 \\ 2 & 4 & 5 & 6 & 0 & 0 \\ 2 & 4 & 5 & 6 & 0 & 0 \\
2 & 5 & 6 & 1 & 0 & 0 \\ 2 & 5 & 6 & 1 & 0 & 0 \\ 2 & 5 & 6 & 1 & 0 & 0 \\
2 & 6 & 1 & 3 & 0 & 0 \\ 2 & 6 & 1 & 3 & 0 & 0 \\ 2 & 6 & 1 & 3 & 0 & 0 \\
3 & 1 & 2 & 4 & 0 & 0 \\ 3 & 1 & 2 & 4 & 0 & 0 \\ 3 & 1 & 2 & 4 & 0 & 0 \\
3 & 2 & 4 & 5 & 0 & 0 \\ 3 & 2 & 4 & 5 & 0 & 0 \\ 3 & 2 & 4 & 5 & 0 & 0 \\
3 & 4 & 5 & 6 & 0 & 0 \\ 3 & 4 & 5 & 6 & 0 & 0 \\ 3 & 4 & 5 & 6 & 0 & 0 \\
3 & 5 & 6 & 1 & 0 & 0 \\ 3 & 5 & 6 & 1 & 0 & 0 \\ 3 & 5 & 6 & 1 & 0 & 0 \\
3 & 6 & 1 & 2 & 0 & 0 \\ 3 & 6 & 1 & 2 & 0 & 0 \\ 3 & 6 & 1 & 2 & 0 & 0
\end{pmatrix}, \mathbf{C}^{(4)}_{\mathbf{45\times 6}} = \begin{pmatrix}
4 & 1 & 2 & 3 & 0 & 0 \\ 4 & 1 & 2 & 3 & 0 & 0 \\ 4 & 1 & 2 & 3 & 0 & 0 \\
4 & 2 & 3 & 5 & 0 & 0 \\ 4 & 2 & 3 & 5 & 0 & 0 \\ 4 & 2 & 3 & 5 & 0 & 0 \\
4 & 3 & 5 & 6 & 0 & 0 \\ 4 & 3 & 5 & 6 & 0 & 0 \\ 4 & 3 & 5 & 6 & 0 & 0 \\
4 & 5 & 6 & 1 & 0 & 0 \\ 4 & 5 & 6 & 1 & 0 & 0 \\ 4 & 5 & 6 & 1 & 0 & 0 \\
4 & 6 & 1 & 2 & 0 & 0 \\ 4 & 6 & 1 & 2 & 0 & 0 \\ 4 & 6 & 1 & 2 & 0 & 0 \\
5 & 1 & 2 & 3 & 0 & 0 \\ 5 & 1 & 2 & 3 & 0 & 0 \\ 5 & 1 & 2 & 3 & 0 & 0 \\
5 & 2 & 3 & 4 & 0 & 0 \\ 5 & 2 & 3 & 4 & 0 & 0 \\ 5 & 2 & 3 & 4 & 0 & 0 \\
5 & 3 & 4 & 6 & 0 & 0 \\ 5 & 3 & 4 & 6 & 0 & 0 \\ 5 & 3 & 4 & 6 & 0 & 0 \\
5 & 4 & 6 & 1 & 0 & 0 \\ 5 & 4 & 6 & 1 & 0 & 0 \\ 5 & 4 & 6 & 1 & 0 & 0 \\
5 & 6 & 1 & 2 & 0 & 0 \\ 5 & 6 & 1 & 2 & 0 & 0 \\ 5 & 6 & 1 & 2 & 0 & 0 \\
6 & 1 & 2 & 3 & 0 & 0 \\ 6 & 1 & 2 & 3 & 0 & 0 \\ 6 & 1 & 2 & 3 & 0 & 0 \\
6 & 2 & 3 & 4 & 0 & 0 \\ 6 & 2 & 3 & 4 & 0 & 0 \\ 6 & 2 & 3 & 4 & 0 & 0 \\
6 & 3 & 4 & 5 & 0 & 0 \\ 6 & 3 & 4 & 5 & 0 & 0 \\ 6 & 3 & 4 & 5 & 0 & 0 \\
6 & 4 & 5 & 1 & 0 & 0 \\ 6 & 4 & 5 & 1 & 0 & 0 \\ 6 & 4 & 5 & 1 & 0 & 0 \\
6 & 5 & 1 & 2 & 0 & 0 \\ 6 & 5 & 1 & 2 & 0 & 0 \\ 6 & 5 & 1 & 2 & 0 & 0
\end{pmatrix}.$$

$$\mathbf{A}^{(5)}_{\mathbf{90\times 6}} = \begin{pmatrix} \mathbf{B}^{(5)}_{\mathbf{45\times 6}} \\ \mathbf{C}^{(5)}_{\mathbf{45\times 6}} \end{pmatrix}, \mathbf{B}^{(5)}_{\mathbf{45\times 6}} = \begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 0 \\ 1 & 2 & 3 & 4 & 5 & 0 \\ 1 & 2 & 3 & 4 & 5 & 0 \\
1 & 3 & 4 & 5 & 6 & 0 \\ 1 & 3 & 4 & 5 & 6 & 0 \\ 1 & 3 & 4 & 5 & 6 & 0 \\
1 & 4 & 5 & 6 & 2 & 0 \\ 1 & 4 & 5 & 6 & 2 & 0 \\ 1 & 4 & 5 & 6 & 2 & 0 \\
1 & 5 & 6 & 2 & 3 & 0 \\ 1 & 5 & 6 & 2 & 3 & 0 \\ 1 & 5 & 6 & 2 & 3 & 0 \\
1 & 6 & 2 & 3 & 4 & 0 \\ 1 & 6 & 2 & 3 & 4 & 0 \\ 1 & 6 & 2 & 3 & 4 & 0 \\
2 & 1 & 3 & 4 & 5 & 0 \\ 2 & 1 & 3 & 4 & 5 & 0 \\ 2 & 1 & 3 & 4 & 5 & 0 \\
2 & 3 & 4 & 5 & 6 & 0 \\ 2 & 3 & 4 & 5 & 6 & 0 \\ 2 & 3 & 4 & 5 & 6 & 0 \\
2 & 4 & 5 & 6 & 1 & 0 \\ 2 & 4 & 5 & 6 & 1 & 0 \\ 2 & 4 & 5 & 6 & 1 & 0 \\
2 & 5 & 6 & 1 & 3 & 0 \\ 2 & 5 & 6 & 1 & 3 & 0 \\ 2 & 5 & 6 & 1 & 3 & 0 \\
2 & 6 & 1 & 3 & 4 & 0 \\ 2 & 6 & 1 & 3 & 4 & 0 \\ 2 & 6 & 1 & 3 & 4 & 0 \\
3 & 1 & 2 & 4 & 5 & 0 \\ 3 & 1 & 2 & 4 & 5 & 0 \\ 3 & 1 & 2 & 4 & 5 & 0 \\
3 & 2 & 4 & 5 & 6 & 0 \\ 3 & 2 & 4 & 5 & 6 & 0 \\ 3 & 2 & 4 & 5 & 6 & 0 \\
3 & 4 & 5 & 6 & 1 & 0 \\ 3 & 4 & 5 & 6 & 1 & 0 \\ 3 & 4 & 5 & 6 & 1 & 0 \\
3 & 5 & 6 & 1 & 2 & 0 \\ 3 & 5 & 6 & 1 & 2 & 0 \\ 3 & 5 & 6 & 1 & 2 & 0 \\
3 & 6 & 1 & 2 & 4 & 0 \\ 3 & 6 & 1 & 2 & 4 & 0 \\ 3 & 6 & 1 & 2 & 4 & 0
\end{pmatrix}, \mathbf{C}^{(5)}_{\mathbf{45\times 6}} = \begin{pmatrix}
4 & 1 & 2 & 3 & 5 & 0 \\ 4 & 1 & 2 & 3 & 5 & 0 \\ 4 & 1 & 2 & 3 & 5 & 0 \\
4 & 2 & 3 & 5 & 6 & 0 \\ 4 & 2 & 3 & 5 & 6 & 0 \\ 4 & 2 & 3 & 5 & 6 & 0 \\
4 & 3 & 5 & 6 & 1 & 0 \\ 4 & 3 & 5 & 6 & 1 & 0 \\ 4 & 3 & 5 & 6 & 1 & 0 \\
4 & 5 & 6 & 1 & 2 & 0 \\ 4 & 5 & 6 & 1 & 2 & 0 \\ 4 & 5 & 6 & 1 & 2 & 0 \\
4 & 6 & 1 & 2 & 3 & 0 \\ 4 & 6 & 1 & 2 & 3 & 0 \\ 4 & 6 & 1 & 2 & 3 & 0 \\
5 & 1 & 2 & 3 & 4 & 0 \\ 5 & 1 & 2 & 3 & 4 & 0 \\ 5 & 1 & 2 & 3 & 4 & 0 \\
5 & 2 & 3 & 4 & 6 & 0 \\ 5 & 2 & 3 & 4 & 6 & 0 \\ 5 & 2 & 3 & 4 & 6 & 0 \\
5 & 3 & 4 & 6 & 1 & 0 \\ 5 & 3 & 4 & 6 & 1 & 0 \\ 5 & 3 & 4 & 6 & 1 & 0 \\
5 & 4 & 6 & 1 & 2 & 0 \\ 5 & 4 & 6 & 1 & 2 & 0 \\ 5 & 4 & 6 & 1 & 2 & 0 \\
5 & 6 & 1 & 2 & 3 & 0 \\ 5 & 6 & 1 & 2 & 3 & 0 \\ 5 & 6 & 1 & 2 & 3 & 0 \\
6 & 1 & 2 & 3 & 4 & 0 \\ 6 & 1 & 2 & 3 & 4 & 0 \\ 6 & 1 & 2 & 3 & 4 & 0 \\
6 & 2 & 3 & 4 & 5 & 0 \\ 6 & 2 & 3 & 4 & 5 & 0 \\ 6 & 2 & 3 & 4 & 5 & 0 \\
6 & 3 & 4 & 5 & 1 & 0 \\ 6 & 3 & 4 & 5 & 1 & 0 \\ 6 & 3 & 4 & 5 & 1 & 0 \\
6 & 4 & 5 & 1 & 2 & 0 \\ 6 & 4 & 5 & 1 & 2 & 0 \\ 6 & 4 & 5 & 1 & 2 & 0 \\
6 & 5 & 1 & 2 & 3 & 0 \\ 6 & 5 & 1 & 2 & 3 & 0 \\ 6 & 5 & 1 & 2 & 3 & 0
\end{pmatrix}.$$

$$\mathbf{A}^{(6)}_{90\times 6} = \begin{pmatrix} \mathbf{B}^{(6)}_{45\times 6} \\ \mathbf{C}^{(6)}_{45\times 6} \end{pmatrix}, \mathbf{B}^{(6)}_{45\times 6} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 4 & 5 & 6 & 2 \\ 1 & 3 & 4 & 5 & 6 & 2 \\ 1 & 3 & 4 & 5 & 6 & 2 \\ 1 & 4 & 5 & 6 & 2 & 3 \\ 1 & 4 & 5 & 6 & 2 & 3 \\ 1 & 4 & 5 & 6 & 2 & 3 \\ 1 & 5 & 6 & 2 & 3 & 4 \\ 1 & 5 & 6 & 2 & 3 & 4 \\ 1 & 5 & 6 & 2 & 3 & 4 \\ 1 & 6 & 2 & 3 & 4 & 5 \\ 1 & 6 & 2 & 3 & 4 & 5 \\ 1 & 6 & 2 & 3 & 4 & 5 \\ 2 & 1 & 3 & 4 & 5 & 6 \\ 2 & 1 & 3 & 4 & 5 & 6 \\ 2 & 1 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 2 & 4 & 5 & 6 & 1 & 3 \\ 2 & 4 & 5 & 6 & 1 & 3 \\ 2 & 4 & 5 & 6 & 1 & 3 \\ 2 & 5 & 6 & 1 & 3 & 4 \\ 2 & 5 & 6 & 1 & 3 & 4 \\ 2 & 5 & 6 & 1 & 3 & 4 \\ 2 & 6 & 1 & 3 & 4 & 5 \\ 2 & 6 & 1 & 3 & 4 & 5 \\ 2 & 6 & 1 & 3 & 4 & 5 \\ 3 & 1 & 2 & 4 & 5 & 6 \\ 3 & 1 & 2 & 4 & 5 & 6 \\ 3 & 1 & 2 & 4 & 5 & 6 \\ 3 & 2 & 4 & 5 & 6 & 1 \\ 3 & 2 & 4 & 5 & 6 & 1 \\ 3 & 2 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 3 & 5 & 6 & 1 & 2 & 4 \\ 3 & 5 & 6 & 1 & 2 & 4 \\ 3 & 5 & 6 & 1 & 2 & 4 \\ 3 & 6 & 1 & 2 & 4 & 5 \\ 3 & 6 & 1 & 2 & 4 & 5 \\ 3 & 6 & 1 & 2 & 4 & 5 \end{pmatrix}, \mathbf{C}^{(6)}_{45\times 6} = \begin{pmatrix} 4 & 1 & 2 & 3 & 5 & 6 \\ 4 & 1 & 2 & 3 & 5 & 6 \\ 4 & 1 & 2 & 3 & 5 & 6 \\ 4 & 2 & 3 & 5 & 6 & 1 \\ 4 & 2 & 3 & 5 & 6 & 1 \\ 4 & 2 & 3 & 5 & 6 & 1 \\ 4 & 3 & 5 & 6 & 1 & 2 \\ 4 & 3 & 5 & 6 & 1 & 2 \\ 4 & 3 & 5 & 6 & 1 & 2 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 4 & 6 & 1 & 2 & 3 & 5 \\ 4 & 6 & 1 & 2 & 3 & 5 \\ 4 & 6 & 1 & 2 & 3 & 5 \\ 5 & 1 & 2 & 3 & 4 & 6 \\ 5 & 1 & 2 & 3 & 4 & 6 \\ 5 & 1 & 2 & 3 & 4 & 6 \\ 5 & 2 & 3 & 4 & 6 & 1 \\ 5 & 2 & 3 & 4 & 6 & 1 \\ 5 & 2 & 3 & 4 & 6 & 1 \\ 5 & 3 & 4 & 6 & 1 & 2 \\ 5 & 3 & 4 & 6 & 1 & 2 \\ 5 & 3 & 4 & 6 & 1 & 2 \\ 5 & 4 & 6 & 1 & 2 & 3 \\ 5 & 4 & 6 & 1 & 2 & 3 \\ 5 & 4 & 6 & 1 & 2 & 3 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 6 & 1 & 2 & 3 & 4 & 5 \\ 6 & 2 & 3 & 4 & 5 & 1 \\ 6 & 2 & 3 & 4 & 5 & 1 \\ 6 & 2 & 3 & 4 & 5 & 1 \\ 6 & 3 & 4 & 5 & 1 & 2 \\ 6 & 3 & 4 & 5 & 1 & 2 \\ 6 & 3 & 4 & 5 & 1 & 2 \\ 6 & 4 & 5 & 1 & 2 & 3 \\ 6 & 4 & 5 & 1 & 2 & 3 \\ 6 & 4 & 5 & 1 & 2 & 3 \\ 6 & 5 & 1 & 2 & 3 & 4 \\ 6 & 5 & 1 & 2 & 3 & 4 \\ 6 & 5 & 1 & 2 & 3 & 4 \end{pmatrix}.$$

Appendix B. Agent-Skill Matrices for the Unbalanced Example in Section 4.

In this appendix we display three of the agent-skill matrices used for the final unbalanced example. Here is the first agent-skill matrix:

$$\mathbf{A}^{(\mathbf{2})}_{\mathbf{90\times 6}} = \begin{matrix}\mathbf{B_{38\times 6}}\\ \mathbf{C_{52\times 6}}\end{matrix}, \mathbf{B_{38\times 6}} = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0\\ 1 & 3 & 0 & 0 & 0 & 0\\ 1 & 4 & 0 & 0 & 0 & 0\\ 1 & 5 & 0 & 0 & 0 & 0\\ 1 & 6 & 0 & 0 & 0 & 0\\ 2 & 1 & 0 & 0 & 0 & 0\\ 2 & 3 & 0 & 0 & 0 & 0\\ 2 & 4 & 0 & 0 & 0 & 0\\ 2 & 5 & 0 & 0 & 0 & 0\\ 2 & 6 & 0 & 0 & 0 & 0\\ 3 & 1 & 0 & 0 & 0 & 0\\ 3 & 2 & 0 & 0 & 0 & 0\\ 3 & 4 & 0 & 0 & 0 & 0\\ 3 & 4 & 0 & 0 & 0 & 0\\ 3 & 5 & 0 & 0 & 0 & 0\\ 3 & 5 & 0 & 0 & 0 & 0\\ 3 & 5 & 0 & 0 & 0 & 0\\ 3 & 6 & 0 & 0 & 0 & 0\\ 3 & 6 & 0 & 0 & 0 & 0\\ 3 & 6 & 0 & 0 & 0 & 0\\ 3 & 6 & 0 & 0 & 0 & 0\\ 3 & 6 & 0 & 0 & 0 & 0\\ 4 & 1 & 0 & 0 & 0 & 0\\ 4 & 2 & 0 & 0 & 0 & 0\\ 4 & 3 & 0 & 0 & 0 & 0\\ 4 & 3 & 0 & 0 & 0 & 0\\ 4 & 3 & 0 & 0 & 0 & 0\\ 4 & 5 & 0 & 0 & 0 & 0\\ 4 & 5 & 0 & 0 & 0 & 0\\ 4 & 5 & 0 & 0 & 0 & 0\\ 4 & 5 & 0 & 0 & 0 & 0\\ 4 & 6 & 0 & 0 & 0 & 0\\ 4 & 6 & 0 & 0 & 0 & 0\\ 4 & 6 & 0 & 0 & 0 & 0\\ 4 & 6 & 0 & 0 & 0 & 0\\ 4 & 6 & 0 & 0 & 0 & 0\\ 4 & 6 & 0 & 0 & 0 & 0\\ 4 & 6 & 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{C_{52\times 6}} = \begin{pmatrix} 5 & 1 & 0 & 0 & 0 & 0\\ 5 & 1 & 0 & 0 & 0 & 0\\ 5 & 2 & 0 & 0 & 0 & 0\\ 5 & 3 & 0 & 0 & 0 & 0\\ 5 & 3 & 0 & 0 & 0 & 0\\ 5 & 3 & 0 & 0 & 0 & 0\\ 5 & 3 & 0 & 0 & 0 & 0\\ 5 & 4 & 0 & 0 & 0 & 0\\ 5 & 4 & 0 & 0 & 0 & 0\\ 5 & 4 & 0 & 0 & 0 & 0\\ 5 & 4 & 0 & 0 & 0 & 0\\ 5 & 4 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 5 & 6 & 0 & 0 & 0 & 0\\ 6 & 1 & 0 & 0 & 0 & 0\\ 6 & 1 & 0 & 0 & 0 & 0\\ 6 & 1 & 0 & 0 & 0 & 0\\ 6 & 2 & 0 & 0 & 0 & 0\\ 6 & 2 & 0 & 0 & 0 & 0\\ 6 & 2 & 0 & 0 & 0 & 0\\ 6 & 3 & 0 & 0 & 0 & 0\\ 6 & 3 & 0 & 0 & 0 & 0\\ 6 & 3 & 0 & 0 & 0 & 0\\ 6 & 3 & 0 & 0 & 0 & 0\\ 6 & 3 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 4 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0\\ 6 & 5 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Here is the second agent-skill matrix for the unbalanced example. The three new rows are highlighted in boldface.

$$\mathbf{A}^{(\mathbf{2})}_{\mathbf{91\times 6}} = \begin{matrix}\mathbf{B_{40\times 6}}\\ \mathbf{C_{51\times 6}}\end{matrix}, \mathbf{B_{40\times 6}} = \begin{pmatrix}
1 & 2 & 0 & 0 & 0 & 0\\
1 & 3 & 0 & 0 & 0 & 0\\
1 & 4 & 0 & 0 & 0 & 0\\
1 & 5 & 0 & 0 & 0 & 0\\
1 & 6 & 0 & 0 & 0 & 0\\
\mathbf{1} & \mathbf{6} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\
2 & 1 & 0 & 0 & 0 & 0\\
2 & 3 & 0 & 0 & 0 & 0\\
2 & 4 & 0 & 0 & 0 & 0\\
2 & 5 & 0 & 0 & 0 & 0\\
2 & 6 & 0 & 0 & 0 & 0\\
\mathbf{2} & \mathbf{6} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\
3 & 1 & 0 & 0 & 0 & 0\\
3 & 2 & 0 & 0 & 0 & 0\\
3 & 4 & 0 & 0 & 0 & 0\\
3 & 4 & 0 & 0 & 0 & 0\\
3 & 5 & 0 & 0 & 0 & 0\\
3 & 5 & 0 & 0 & 0 & 0\\
3 & 5 & 0 & 0 & 0 & 0\\
3 & 6 & 0 & 0 & 0 & 0\\
3 & 6 & 0 & 0 & 0 & 0\\
3 & 6 & 0 & 0 & 0 & 0\\
3 & 6 & 0 & 0 & 0 & 0\\
3 & 6 & 0 & 0 & 0 & 0\\
4 & 1 & 0 & 0 & 0 & 0\\
4 & 2 & 0 & 0 & 0 & 0\\
4 & 3 & 0 & 0 & 0 & 0\\
4 & 3 & 0 & 0 & 0 & 0\\
4 & 3 & 0 & 0 & 0 & 0\\
4 & 5 & 0 & 0 & 0 & 0\\
4 & 5 & 0 & 0 & 0 & 0\\
4 & 5 & 0 & 0 & 0 & 0\\
4 & 5 & 0 & 0 & 0 & 0\\
4 & 6 & 0 & 0 & 0 & 0\\
4 & 6 & 0 & 0 & 0 & 0\\
4 & 6 & 0 & 0 & 0 & 0\\
4 & 6 & 0 & 0 & 0 & 0\\
4 & 6 & 0 & 0 & 0 & 0\\
4 & 6 & 0 & 0 & 0 & 0
\end{pmatrix}, \mathbf{C_{51\times 6}} = \begin{pmatrix}
5 & 1 & 0 & 0 & 0 & 0\\
5 & 1 & 0 & 0 & 0 & 0\\
5 & 2 & 0 & 0 & 0 & 0\\
\mathbf{5} & \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\
5 & 3 & 0 & 0 & 0 & 0\\
5 & 3 & 0 & 0 & 0 & 0\\
5 & 3 & 0 & 0 & 0 & 0\\
5 & 3 & 0 & 0 & 0 & 0\\
5 & 4 & 0 & 0 & 0 & 0\\
5 & 4 & 0 & 0 & 0 & 0\\
5 & 4 & 0 & 0 & 0 & 0\\
5 & 4 & 0 & 0 & 0 & 0\\
5 & 4 & 0 & 0 & 0 & 0\\
5 & 6 & 0 & 0 & 0 & 0\\
5 & 6 & 0 & 0 & 0 & 0\\
5 & 6 & 0 & 0 & 0 & 0\\
5 & 6 & 0 & 0 & 0 & 0\\
5 & 6 & 0 & 0 & 0 & 0\\
5 & 6 & 0 & 0 & 0 & 0\\
5 & 6 & 0 & 0 & 0 & 0\\
5 & 6 & 0 & 0 & 0 & 0\\
6 & 1 & 0 & 0 & 0 & 0\\
6 & 1 & 0 & 0 & 0 & 0\\
6 & 1 & 0 & 0 & 0 & 0\\
6 & 2 & 0 & 0 & 0 & 0\\
6 & 2 & 0 & 0 & 0 & 0\\
6 & 2 & 0 & 0 & 0 & 0\\
6 & 3 & 0 & 0 & 0 & 0\\
6 & 3 & 0 & 0 & 0 & 0\\
6 & 3 & 0 & 0 & 0 & 0\\
6 & 3 & 0 & 0 & 0 & 0\\
6 & 3 & 0 & 0 & 0 & 0\\
6 & 4 & 0 & 0 & 0 & 0\\
6 & 4 & 0 & 0 & 0 & 0\\
6 & 4 & 0 & 0 & 0 & 0\\
6 & 4 & 0 & 0 & 0 & 0\\
6 & 4 & 0 & 0 & 0 & 0\\
6 & 4 & 0 & 0 & 0 & 0\\
6 & 4 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0\\
6 & 5 & 0 & 0 & 0 & 0
\end{pmatrix}.$$

Here is the agent-skill matrix associated with the final feasible solution $(91, 20)$:

$$\mathbf{A}^{(\mathbf{2})}_{\mathbf{91\times 6}} = \begin{matrix}\mathbf{B_{42\times 6}}\\ \mathbf{C_{49\times 6}}\end{matrix}, \mathbf{B_{42\times 6}} = \begin{pmatrix}
1&2&0&0&0&0\\1&2&0&0&0&0\\1&3&0&0&0&0\\1&4&0&0&0&0\\1&5&0&0&0&0\\1&6&0&0&0&0\\1&6&0&0&0&0\\
2&1&0&0&0&0\\2&1&0&0&0&0\\2&3&0&0&0&0\\2&4&0&0&0&0\\2&5&0&0&0&0\\2&6&0&0&0&0\\2&6&0&0&0&0\\
3&1&0&0&0&0\\3&2&0&0&0&0\\3&4&0&0&0&0\\3&4&0&0&0&0\\3&4&0&0&0&0\\3&5&0&0&0&0\\3&5&0&0&0&0\\3&5&0&0&0&0\\
3&6&0&0&0&0\\3&6&0&0&0&0\\3&6&0&0&0&0\\3&6&0&0&0&0\\3&6&0&0&0&0\\
4&1&0&0&0&0\\4&2&0&0&0&0\\4&3&0&0&0&0\\4&3&0&0&0&0\\4&3&0&0&0&0\\4&5&0&0&0&0\\4&5&0&0&0&0\\4&5&0&0&0&0\\4&5&0&0&0&0\\
4&6&0&0&0&0\\4&6&0&0&0&0\\4&6&0&0&0&0\\4&6&0&0&0&0\\4&6&0&0&0&0\\4&6&0&0&0&0
\end{pmatrix}, \mathbf{C_{49\times 6}} = \begin{pmatrix}
5&1&0&0&0&0\\5&1&0&0&0&0\\5&2&0&0&0&0\\5&2&0&0&0&0\\5&3&0&0&0&0\\5&3&0&0&0&0\\5&3&0&0&0&0\\5&3&0&0&0&0\\
5&4&0&0&0&0\\5&4&0&0&0&0\\5&4&0&0&0&0\\5&4&0&0&0&0\\5&4&0&0&0&0\\5&6&0&0&0&0\\5&6&0&0&0&0\\5&6&0&0&0&0\\
5&6&0&0&0&0\\5&6&0&0&0&0\\5&6&0&0&0&0\\5&6&0&0&0&0\\5&6&0&0&0&0\\
6&1&0&0&0&0\\6&1&0&0&0&0\\6&1&0&0&0&0\\6&2&0&0&0&0\\6&2&0&0&0&0\\6&2&0&0&0&0\\6&3&0&0&0&0\\6&3&0&0&0&0\\
6&3&0&0&0&0\\6&3&0&0&0&0\\6&3&0&0&0&0\\6&4&0&0&0&0\\6&4&0&0&0&0\\6&4&0&0&0&0\\6&4&0&0&0&0\\6&4&0&0&0&0\\
6&4&0&0&0&0\\6&4&0&0&0&0\\6&4&0&0&0&0\\6&5&0&0&0&0\\6&5&0&0&0&0\\6&5&0&0&0&0\\6&5&0&0&0&0\\6&5&0&0&0&0\\
6&5&0&0&0&0\\6&5&0&0&0&0\\6&5&0&0&0&0
\end{pmatrix}.$$