# RESOURCE POOLING AND STAFFING IN CALL CENTERS WITH SKILL-BASED ROUTING

by

Rodney B. Wallace

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

Ward Whitt

Columbia University
ward.whitt@columbia.edu

March 12, 4004, Revision: July 13, 2004

*Abstract*

Call centers usually handle several types of calls. To do so effectively, agents are given special training. However, it is often not 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. Call centers are equipped with automatic call distributors to assign calls to appropriate agents, i.e., to perform skill-based routing (SBR). However, it is challenging to do skill-based routing well. Moreover, it is challenging to determine the staff requirements in an SBR call center. This paper addresses these problems. The main idea is to seek simplification through resource pooling, i.e., the notion that with (i) only a limited amount of cross training (e.g., perhaps even when agents have at most two skills)and (ii) a reasonable scheduling policy, the call center performance may be nearly the same as if all agents had all skills (and scheduling was not an issue at all). First, simulation experiments within a particular SBR framework show that resource pooling indeed occurs. Second, resource pooling is exploited to develop an effective algorithm to generate staffing requirements. The staffing algorithm exploits the classical Erlang model to determine an initial estimate for the total staff needed in the call center. Motivated by heavy-traffic stochastic-process limits, a square-root formula is then applied to determine initial primary-skill requirements within the estimated total staff. Then a fair-assignment scheme is developed to allocate additional skills. Finally, simulations are performed in order to make improvements in the initial assignment, in order to make the total staff requirements as small as possible, while ensuring that all performance requirements are met. Simulation experiments show that the overall procedure can be remarkably effective: The required staff with limited cross training in a reasonable SBR framework can be nearly the same as if all agents had all skills.

*Subject classifications:* Queues, applications: resource pooling in call centers. Queues, algorithms: staffing in call centers with skill-based routing. Queues, networks: Call centers with skill-based routing.

*Area of Review:* Stochastic Models.

*Keywords*: telephone call centers, customer contact centers, staffing, scheduling, skill-based routing, resource pooling, queues, multi-server queues, multi-class queues, simulation.

## 1. Introduction

The purpose of this paper is to provide insights and methods to help improve the design and management of telephone *call centers* and more general customer contact centers allowing contact through other media such as email. As indicated in the review by Gans, Koole and Mandelbaum (2003), systematic analysis is needed because call centers have become quite complicated.

**Skill-Based Routing.** Call centers usually handle several types of calls. However, it is usually not possible or cost-effective to train every agent (customer service representative) to be able to handle every type of call. For example, with the globalization of many businesses, call centers often receive calls in several different languages. The callers and agents each may speak one or more of several possible languages, but not necessarily all of them. And, of course, it may simply not be possible for the agents to learn all the languages. But it may well be possible to find agents that can speak two or three languages.

Another classification of calls involves special promotions. The callers may be calling a special 800 number designated for the special promotion. Agents typically are trained to respond to inquiries about some of the promotions, but not all of them. Learning about special promotions is certainly less difficult than learning entire languages, but it tends to be prohibitive to train all agents to be able to respond to calls about all promotions, especially when there is a very short time span between the creation and the delivery of the promotion.

Specialization also naturally arises in *strategic outsourcing*, where one company turns over some of its business functions to a third-party contractor. An example is the management of the company's computer and information systems. The strategic outsourcing is usually managed by a *single-point-of-contact help desk*, which in fact is a call center. In these technical help desks, agents (service representatives) may only be able to help customers with some of their technical problems. The agents may have different levels of skills, with some skills requiring extensive training.

Thus, frequently, the calls have different requirements and the agents have different skills. Fortunately, modern *automatic call distributors* (ACD's) have the capability of assigning calls to agents with the appropriate skills. Thus the call-center information-and-communication-technology (ICT) equipment can allow for the generalization to multiple call types. That capability is called *skill-based routing* (SBR).

Most current call centers perform skill-based routing, and many do so remarkably well. Nevertheless, there remains a need for fundamental principles and operations-research techniques that will make it possible to better design and manage SBR call centers.

In fact, skill-based routing is now an active research topic, but so far there is only a relatively small body of literature. For example, Garnett and Mandelbaum (2000) conducted a simulation study to show some of the operational complexities of skill-based routing. Research also has been done by Perry and Nilsson (1992), Koole and Talim (2000) and Borst and Seri (2000). An interesting fluid-model approach to staffing in SBR call centers has recently been proposed by Harrison and Zeevi (2003). Our intent is to contribute to this line of research.

**Resource Pooling.** We propose a two-word answer to the SBR problem: *resource pooling.* Within the context of an SBR call center, resource pooling means that with a limited amount of cross training (having agents with multiple skills) and a reasonable scheduling algorithm, a diverse SBR call center may perform nearly the same as if all agents had all skills (and scheduling were not an issue at all).

The resource-pooling notion is important, because it may lead to much greater efficiency, with significant performance benefits and cost reductions. Of course, resource pooling is not the solution to all problems, but we intend to show that it is an important concept for SBR call centers. In particular, if there is inadequate sharing of skills, then the SBR call center will perform nearly the same as several separate smaller call centers, and thus lose important economies of scale. If the separate smaller groups are themselves still large, then little efficiency will be lost, but if the separate groups are indeed not large, then the performance can be much worse than if all agents had all skills.

We have noted that performance may be much better if all agents are *universal agents*, i.e., if all agents have all skills, than if the agents each have only one skill. However, previously we gave compelling examples (e.g., languages and computer skills) showing that it is often not feasible for many agents to have all skills. The main point, then, is that *a little flexibility may go a long way*: We will show that an SBR call center can perform nearly as well as an SBR call center staffed by universal agents (where all agents have all skills) if the agents only have relatively few skills, in particular, only two skills.

In particular, in this paper we describe simulation experiments conducted in order to investigate resource pooling in SBR call centers. In these simulation experiments we consider a call center with 90 agents and 6 call types. In six separate simulation runs, we then see what happens if all agents have $j$ skills, $1 \leq j \leq 6$. Consistent with the remark above, first we show that the performance is much better for $j = 6$ than for $j = 1$, demonstrating the potential importance of resource pooling. As should be expected, the performance improves as $j$ increases. However, we show that the performance with $j = 2$ is much closer to the performance with $j = 6$ than the performance with $j = 1$. Indeed, the performance with $j = 2$ is almost as good as the performance with $j = 6$. It is natural to contemplate more cases, e.g., in which only some of the agents have two skills, but we believe our experiments adequately address the main point.

**Implications.** No doubt, the main implication of these experimental results is the importance of paying careful attention to the agent skill mix. In most discussions of SBR call centers, the skill mix is taken as given, but in fact in the design and operation of an SBR call center, there are opportunities to determine the agent skill mix. Our experiments show that there can be great benefit in having some flexibility (agents having multiple skills), but the value added by having great flexibility (agents having a large number of skills) is likely not to be so great. We believe that is a fundamental principle that can greatly help call-center managers.

Moreover, resource pooling has strong implications about the importance of skill-based routing algorithms. If indeed limited flexibility within a reasonable, but suboptimal, SBR framework can produce performance nearly as good as if all agents had all skills, then we will have succeeded in circumventing the hard scheduling problem. If we can achieve the desired performance benefit through limited flexibility, it may be unnecessary to address the difficult scheduling problem.

Of course, it remains to investigate the extent to which the "great benefit of limited flexibility" principle is applicable in different call-center settings. We would suggest that this principle might even be used to evaluate alternative call-center designs. In particular, we contend that one scheme for operating an SBR call center might well be preferred to another if it better facilitates great benefits from limited flexibility.

Another implication is that, if we design and manage the SBR call center so that there is sufficient flexibility to support resource pooling, then the SBR call center will be easier to operate. To firmly support that idea, in this paper we show how to exploit resource pooling in order to develop an algorithm to determine staff (agents) and equipment (telephone trunk lines) requirements in an SBR call center. To do so, we specify a specific framework to implement skill-based routing. Then, within that framework, we develop an algorithm for staff and equipment provisioning. A key idea is to choose a reasonable SBR framework, rather than search for an optimal scheduling policy.

The centerpiece of our provisioning algorithm is the resource-pooling notion. In particular, in the initial phase of our provisioning algorithm, when we start to decide on the total number of agents to be in the call center, we assume that resource pooling will ultimately be present, and thus we apply the classical Markovian $M/M/C/K$ model, acting as if skills do not matter. With that simplification, the optimal number of agents can easily be found by applying fast numerical algorithms, e.g., as in Borst, Mandelbaum and Reiman (2004), Massey and Wallace (2004) and Whitt (2003). (The algorithm in Whitt (2003) allows the treatment of customer abandonment with non-exponential service-time and abandon-time distributions, but in this paper we do not consider those features. Those extensions are known to be important in practice; e.g., see Brown et al. (2002). The algorithm in Jennings et al. (1996) could be used to address time-dependent arrivals, which we do not consider here either.)

But we need to go further: After specifying the total number of agents required, we determine the specific skill requirements for these agents. We thus generate a full specification of the agent-skill requirements, which can be used as input to the staff scheduling phase of call-center management. (Recall that call centers typically use a four-phase process to determine and manage staffing levels. The first phase involves forecasting customer arrival rates and expected service times across the day. The second phase uses these forecasts to determine the staff and equipment requirements. The third phase uses the output of the second phase, together with appropriate optimization tools, to determine what equipment is needed and make detailed staff schedules, allowing for meals and breaks. Finally, the fourth phase involves real-time adjustments during service delivery. In this paper, we are only considering the second phase.)

Next, given a fully specified provisioning, we employ simulation to make small local refinements in the initial provisioning, aiming to reduce the required staff while meeting specified constraints on performance. In applying simulation to analyze call-center models, we are following common practice; e.g., see Anton et al. (1999) and Brigandi et al. (1994). In seeking improved polices by performing local search, we are following Choudhury et al. (1995); there the algorithm was numerical transform inversion instead of simulation. The resource pooling helps us quickly find a good starting point for the second, simulation phase of our provisioning algorithm.

As we apply simulation to improve our agent-skill-matrix assignment, we see the realized performance. Thus we can directly verify that our proposed provisioning algorithm is indeed effective. And, indeed, these simulation experiments provide additional support for the resource-pooling notion, because the algorithm shows that the number of agents and telephone trunk lines required in the SBR call center, where each agent has only two skills, is nearly the same as if each agent had all skills. In other words, not only is the performance for given provisioning nearly the same as when all agents have all skills (as shown in Section 3), but also the provisioning for given performance requirements is nearly the same as when all agents have all skills (shown in Sections 5 and 6). *Amazingly, in our examples, there is only a difference of a single agent*: In our examples, 90 agents suffice when all agents have all skills, while 91 agents suffice when all agents have at most two skills. We do not claim that the same spectacular resource-pooling benefit necessarily will hold in all other settings, but the examples here show the potential.

**A Little Flexibility Goes a Long Way.** The conclusions here have implications for other contexts involving decision-making under uncertainty, suggesting that we might expect a little flexibility to go a long way. On the other hand, it should come as no surprise that this important idea has already been advanced in other contexts. The idea is natural, so it is hard to properly trace its history, but we can cite several instances closely related to the call-center context.

Significant theoretical work exposing the advantage of a small amount of flexibility was done by Azar et al. (1994), Vvedenskaya et al. (1996), Turner (1996) and Mitzenmacher (1996). To convey the idea, will will describe the queueing problem considered in Vvedenskaya et al. (1996). It involves assigning arriving customers, immediately upon arrival, to one of several identical queues. In the model there is a single stream of customers arriving in a Poisson process. Upon arrival, each customer must join one of a large number of independent, identical single-server queues with exponential service times and unlimited waiting space.

The standard approach to this queue-joining problem is to examine the state of all the queues and join one of the queues with the fewest customers. Indeed, Winston (1977) proved that the join-the-shortest-line rule is optimal. (Surprisingly, perhaps, optimality can cease if some of the assumptions are changed, e.g., if the service-time distribution need not be exponential; see Whitt (1986).) A difficulty with the join-the-shortest-queue rule, however, is that it may require obtaining a large amount of state information. In particular, we need to know the number of customers in each of the queues, which may be difficult if there are many widely-distributed queues.

So it is natural to consider alternative approaches that require less information. A simple alternative is to use no information at all. One way to do so is to assign each arrival randomly to one of the queues, with each server being selected having equal chance. However, a random assignment makes the individual servers act as separate $M/M/1$ queues. If we assign successive arrivals randomly to one of the servers, then we lose all the efficiency of the multiserver system.

The key idea in Azar et al. (1994), Vvedenskaya et al. (1996), Turner (1996) and Mitzenmacher (1996) is to allow just a little choice. In particular, it suffices to compare just two queues: An alternative "lightweight" approach is to pick two of the queues at random and join the shorter of those two. The remarkable fact is that this lightweight "low-information" decision rule does almost as well as the join-the shortest-queue rule, requiring full information, which in turn does nearly as well as the large combined group of servers with a single queue, serving in first-come first-served order. That conclusion was formalized by considering the asymptotic behavior as the number of servers gets large (with the arrival rate kept proportional).

A related, alternative scheme, is to pay little attention upon arrival, but then later perform periodic load balancing. The performance of periodic load balancing is studied in Hjlmtsson and Whitt (1998).

Thus, previous mathematical analysis suggests that a little flexibility may go a long way. Thus it is natural to investigate resourse pooling. Related research on resource pooling (with precise meaning depending upon the context) has been done by Mandelbaum and Reiman (1998), Mitzenmacher (1997), Mitzenmacher and Vöcking (1999), Harrison and Lopez (1999), Turner (1999) and Williams (2000). In the same spirit is recent work on cross training of workers; see Andradóttir et al. (2001), Hopp and van Oyen (2003) and references therein.

**Organization of This Paper.** Here is how the rest of this paper is organized: In Section 2 we specify our call-center model; i.e., we specify precisely how we will implement skill-based routing. Next, in Section 3 we describe our experiment to investigate resource pooling in an SBR call center. In Section 4 we present our staff-and-equipment provisioning algorithm, which is in the setting of Section 2. In Sections 5 and 6 we describe simulation experiments to see how the algorithm performs. The first experiment in Section 5 is for a balanced call center, like the one considered in Section 3. The second experiment in Section 6 is for a more realistic unbalanced call center. Additional details can be found in Wallace (2004), on which this paper draws.

## 2. The Call-Center Model

In this section we specify the call-center model that we will consider. There are two parts to the model: (i) a specification of the way that the call center operates, and (ii) stochastic assumptions to be used in our simulation experiments. At the outset we should point out that neither part of the model is fully general. In particular, we introduce a structured framework for the operation of the call center which makes it relatively easy to implement skill-based routing. No doubt, this scheme does not yield optimal routing, even under the strong stochastic assumptions we make, but we believe it is a reasonable framework. It is based on existing commercial systems for implementing skill-based routing. However, as indicated in the introduction, we believe that the main ideas in this paper can be applied more generally.

Our call-center model will be a multi-server queueing system with $C$ servers, $K$ extra waiting spaces and $n$ call types. The pair $(C, K)$ corresponds to having $C$ agents or CSR's and $C + K$ available telephone trunk lines. A call uses a trunk line both while it is waiting and while it is being served. We assume that the ACD can accommodate $C + K$ calls, with any calls not being served waiting. Calls arriving when all $C + K$ trunk lines are occupied are assumed to be blocked and lost.

Now we start introducing our special SBR framework: Each agent can have from 1 to $n$ skills, at distinct levels, ranging from 1 to $n$, where $n$ is the number of call types. Agents can serve a call type if they have the skill for that call type; they cannot serve a call type if they do not have the required skill. In subsections below we specify: (i) how we assign skills and skill levels to agents, (ii) what to do when a new arrival occurs, (iii) what to do when an agent becomes free and (iv) the stochastic assumptions we make in our simulation experiments.

**Assigning Skills to Agents** We specify the skills of agents via a $C \times n$ *agent-skill matrix* $A$. The rows of $A$ correspond to the agents, while the columns of $A$ correspond to *skill levels.* The skill levels are used to assign priorities among different agents possessing a designated skill. (An agent has a skill if and only if he has that skill at some positive level. Lower skill levels have preference in the priority scheme.) We assume that each agent has at most one skill at any given skill level, so that the matrix $A$ can indeed be used to assign skills. Entry $A_{i,j}$ specifies the skill of the $i$th agent at the $j$th skill level, with a 0 indicating no skill at that level.

Row $i$ of $A$ specifies the skills and skill levels for the $i^{\text{th}}$ agent. The first column of $A$ specifies the primary skills. Thus $A_{i,1}$ is the primary skill of agent $i$. We assume that every agent has a primary skill, so that $A_{i,1}$ is an integer with $1 \leq A_{i,1} \leq n$ for each $i$. We group the agents by their primary skills, so work group $G_k$ is the subset of all agents with primary skill $k$, i.e.,

$$G_k \equiv \{i : 1 \leq i \leq C, \quad A_{i,1} = k\}, \quad 1 \leq k \leq n \ .$$

By our assumptions, the collection of work groups is a partition of the set of all agents, i.e., of the set $\{1, \ldots, C\}$. Each agent belongs to one and only one work group. The number of elements in work group $k$ is the number of elements in $G_k$, denoted by $|G_k|$, i.e.,

$$C_k \equiv |G_k| = \sum_{i=1}^{C} 1_{\{A_{i,1}=k\}} \ ,$$

where $1_B$ is the indicator function of the set $B$; i.e., $1_B(i) = 1$ if $i \in B$ and $1_B = 0$ otherwise. We will assume that there is a work group associated with each call type, although that assumption could be relaxed.

The $j^{\text{th}}$ column of $A$ specifies the skills at the $j^{\text{th}}$ skill level. Thus, as indicated above, entry $A_{i,j}$ specifies the skill of the $i^{\text{th}}$ agent at the $j^{\text{th}}$ skill level. If agent $i$ has skill $k$ at level $j$, then $A_{i,j} = k$; if agent has no skill at level $j$, then $A_{i,j} = 0$. We assume that each agent skill appears at most at one level, so no positive number appears more than once in any row. (A second appearance in any row would be redundant, because the higher skill level would dominate.) Thus the positive entries of the $i^{\text{th}}$ row are the skills possessed by the $i^{\text{th}}$ agent. The columns where those positive entries appear indicate the skill levels of those skills for that particular agent. Two agents have the same skills at the same levels if, and only if, their rows in the skill matrix $A$ are identical.

In order to clarify the rules for assigning skills to agents, we consider the following four examples:

$$\mathbf{A_{5\times 1}} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \ \mathbf{A_{3\times 2}} = \begin{pmatrix} 1 & 0 \\ 2 & 0 \\ 2 & 0 \end{pmatrix}, \ \mathbf{A_{4\times 2}} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 2 & 1 \\ 2 & 1 \end{pmatrix}, \ \mathbf{A_{6\times 4}} = \begin{pmatrix} 3 & 4 & 1 & 0 \\ 1 & 4 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ 3 & 1 & 2 & 4 \\ 1 & 0 & 4 & 0 \end{pmatrix}$$

The first matrix $\mathbf{A_{5\times 1}}$ specifies an agent profile for a call center manned by 5 agents, all possessing the same single skill. The second matrix $\mathbf{A_{3\times 2}}$ specifies an agent profile for a call center with 3 agents, each possessing one of two primary skills. The zeros in the second column indicate that no agent has a secondary skill. Thus, this call center has two separate work groups that function as separate call centers: Agent 1 will handle all type-1 calls, while agents 2 and 3 will handle all type-2 calls.

The third matrix $\mathbf{A_{4\times 2}}$ is in the spirit of the bilingual call-center model used in Green (1985) and Stanford and Grassmann (1993, 2000). In this matrix, the first two rows represent 2 agents each with a primary skill to support call type 1 and no secondary skill. These 2 agents form work group one and are referred to as the limited-use or restricted-use servers in Green (1985) and the unilingual group in Stanford and Grassmann (2000). The third and fourth rows represent agents 3 and 4. These agents each have a primary skill to support call type 2 and a secondary skill to support call type 1. They make up work group two and are referred to as the general-use servers and bilingual group in Green (1985) and Stanford and Grassmann (2000), respectively.

In the last example, the matrix $\mathbf{A_{6\times 4}}$ illustrates the more general structure possible for the agent-skill matrix. This example has 6 agents and 4 call types. There are four work groups, one for each call type. Using the first column, we can identify the agent's work group. Work group 1 consists of agents 2 and 6, while work group 2 consists of agent 3. All agents have secondary skills with the exception of agents 4 and 6. Agent 4 supports only call type 4, while agent 5 is a *universal agent*, because he can support all of the call types. Agent 6 can support 2 different service requests: call type 1 at the primary level and call type 4 at the tertiary level. Note that agent 6 has a tertiary skill but no secondary skill.

**What to do when an Arrival Occurs** In order to implement skill-based routing, we need to specify the decisions we will make in two situations: (i) when an arrival occurs, and (ii) when an agent becomes free. We treat each in turn.

Arriving calls of type $k$ are first routed to available agents in work group $k$, because those agents have primary skill $k$ ($G_k = \{i : A_{i,1} = k\}$). The *longest-idle-agent-routing (LIAR) policy* is used to determine which of the idle agents in work group $k$ is to handle the call. The LIAR policy sends the call to the agent in work group $k$ that has been idle the longest since the completion of their last job. The LIAR policy is deemed fair, because it tends to balance the workload across agents. However, other tie-breaking schemes schemes could be used instead. For example, one might choose the idle agent whose cumulative idle time over the last half hour is greatest.

If all agents with call type $k$ as a primary skill are busy, then the call is routed to available agents having call type $k$ as a secondary skill. Again, among all agents having type $k$ as a secondary skill, the LIAR policy is used to pick the actual agent to handle the call. If all agents with call type $k$ as a primary skill or a secondary skill are busy, then the call is routed to available agents having call type $k$ as a tertiary skill, and so on. Again, the LIAR policy is used to break ties.

If no available qualified agent can be found to handle the type-$k$ call immediately upon arrival, then the type-$k$ call is placed on the end of a queue associated with work group $k$.

**What to do when an Agent becomes Free** We now specify what an agent does when he completes handling a call and becomes free. First, if there are no customers in the $n$ queues, then the agent goes idle. Otherwise, the agent visits the queues of the work groups for which he has skills. If there are no waiting calls for which he has skills, then again the agent goes idle. The agent visits the queues in order of the agent's skill levels; i.e., the agent goes first to the queue with his primary skill, second to the queue with his secondary skill, and so forth. The agent serves the call that is first in line in the first nonempty queue in the skill-level order. Thus the agent serves calls within each work group (equivalently, within each call type) in a *first-come first-served* (FCFS) order or, equivalently, in a first-in first-out (FIFO) order.

**Stochastic Assumptions** We now specify stochastic assumptions to be used in our simulation experiments. We make relatively simple assumptions, but we do not believe they are critical to our conclusions.

In our simulation model we assume that $n$ types of calls arrive at the call center according to $n$ mutually independent Poisson processes with rates $\lambda_i$, $1 \leq i \leq n$. It is well known that is equivalent to assuming that all calls arrive according to a single Poisson process with rate $\lambda = \lambda_1 + \cdots + \lambda_n$ and, afterwards, are assigned to be type $i$ with probability $p_i \equiv \lambda_i/\lambda$, according to mutually independent trials, independent of the single Poisson process.

We assume that the call holding (service) times mutually independent exponential random variables, which are independent of the arrival process. We assume that calls of type $k$ have mean $1/\mu_k$, regardless of where they are served. Thus we are assuming that time differences in call processing among different agents can be ignored, given that the agent actually has the required skill. In our simulation experiments, we will go further and assume that all service times have the same mean.

In this paper we do not consider either retrials or abandonments. Thus, calls that are blocked do not affect future arrivals. We assume that service quality is sufficiently high that abandonments can be ignored. Thus we assume that all admitted calls will eventually be served. Again, we believe our conclusions do not depend strongly on these assumptions, but that remains to be verified. Our provisioning algorithm extends to cover abandonments and non-exponential service-time and abandon-time distributions, as we will explain in Section 4.

## 3. 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 to achieve the benefits of resource pooling. We consider a call center serving 6 call types and see what happens when all agents possess $j$ skills, allowing $j$ to range from 1 to 6 in separate simulation runs. When $j = 6$, all agents have all skills, and the call center behaves like a single-group call center. In contrast, when $j = 1$, the call center behaves much like 6 separate call centers, for which the performance is much worse. When $j = 1$, the call center does not behave *exactly* like separate call centers because of the finite waiting room. Here, when $j = 1$, the six call types share the common waiting room. The shared waiting room leads to much better performance than if the waiting room too was divided into segregated portions without any sharing.

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

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 $j \geq 2$, there will be exactly $15/5 = 3$ agents with each combination of the possible primary and secondary skills.

Recall that the offered load with common 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 traffic intensities yielding normal, light and heavy loading depend on the number of servers, increasing as the number of servers increases. The finite waiting room makes it possible to have traffic intensities greater than 1, as would the presence of customer abandonment.

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 also 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 and in Tables 1–3. Figure 1 shows 9 individual graphs in a $3 \times 3$ 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. The six agent-skill matrices are displayed in Appendix A.

Figure 1 dramatically shows the resource pooling. In all three offered-load scenarios, there are significant improvements in performance when agents are given at least two skills. Moreover, 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.

More detailed descriptions of system performance in the 18 cases are shown in Tables 1–3. In these tables we do not display confidence intervals, but the statistical precision can be seen from the results for different work groups. Since the model is perfectly symmetric for $j = 1$, $j = 2$ and $j = 6$, the actual theoretical (infinite-sample) per-work-group performance measures are the same for all groups. Thus the observed variability in the statistics provides a practical estimate of the statistical precision.

In order to put Tables 1–3 into perspective, it is useful to analyze the extreme cases analytically. First, when $j = 6$, the model reduces to three cases of a $M/M/90/30$ model with $1/\mu = 10$ using a time scale of minutes, which we can solve analytically. The three cases are: $\lambda = 8.4$ (normal load), $\lambda = 7.74$ (light load) and $\lambda = 9.0$ (heavy load). Numerical analytical results for these three $M/M/90/30$ models appear in Table 4. These results confirm the six-skill case of the simulation runs in Tables 1–3.

As noted above, when $j = 1$, 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 $j = 1$ 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. With the $M/M/15/5$ model, we again have a time scale of minutes and with $1/\mu = 10$, but now the three possible arrival rates are: $\lambda = 1.4$ (normal load), $\lambda = 1.29$ (light load) and $\lambda = 1.5$ (heavy load). The performance results appear in Table 5. From Tables 5 and 1–3, we see that there is a wide range of performance, none of which is consistently good.

## 4. A Heuristic SBR Provisioning Algorithm

In this section we apply the resource-pooling property to develop an algorithm to generate requirements for staff and trunk lines in our call-center model with skill-based routing. That means that we aim to determine the parameters $C$ and $K$ in the $M_n/M/C/K/SBR$ model and the agent-skill matrix $A$, given the other model parameters and various constraints on $A$ and on performance. Our main idea is to assume that there will be sufficient cross training so that resource pooling holds, but we use simulation to verify that the performance is indeed satisfactory, and to make appropriate adjustments if it is not.

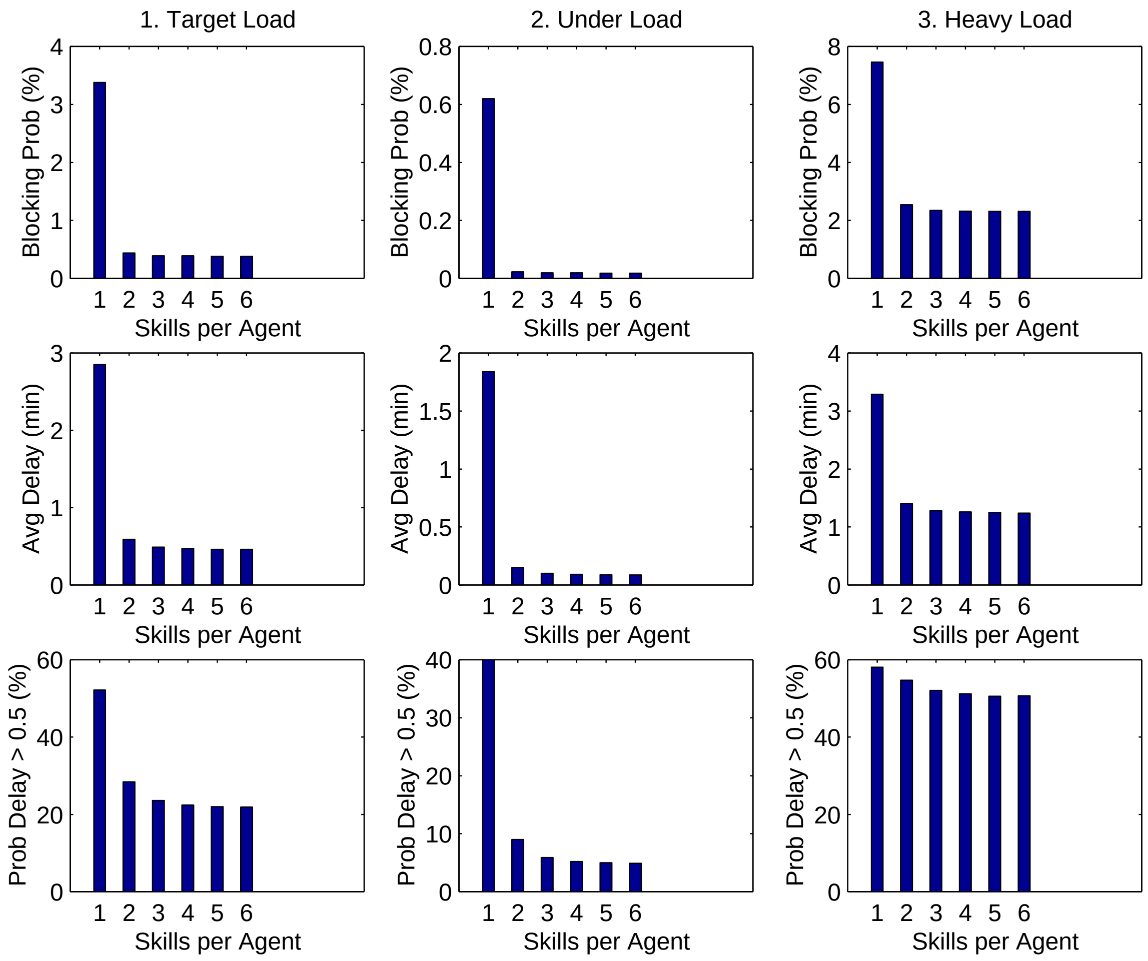

Figure 1: Typical performance measures as a function of the number of skills per agent and the offered load for an $M_6/M/90/30$ SBR call center with $1/\mu_1 = \cdots = 1/\mu_6 = 10$ minutes. The specific performance measures are the blocking probability, the mean conditional delay given entry, and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

| **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. $\mathcal{P}(\text{Delay} \leq 0.5 \,\vert \text{entry})$ (%) | 47.8 | 71.6 | 76.4 | 77.6 | 78.0 | 78.1 |
| 5. $\mathcal{P}(\text{Delay}_1 \leq 0.5 \,\vert \text{entry})$ | 47.3 | 71.1 | 76.4 | 77.5 | 78.0 | 77.9 |
| 5. $\mathcal{P}(\text{Delay}_2 \leq 0.5 \,\vert \text{entry})$ | 47.2 | 71.5 | 76.2 | 77.6 | 77.9 | 78.0 |
| 5. $\mathcal{P}(\text{Delay}_3 \leq 0.5 \,\vert \text{entry})$ | 48.3 | 72.1 | 76.7 | 77.9 | 78.2 | 78.4 |
| 5. $\mathcal{P}(\text{Delay}_4 \leq 0.5 \,\vert \text{entry})$ | 48.7 | 72.2 | 76.6 | 77.6 | 78.1 | 78.1 |
| 5. $\mathcal{P}(\text{Delay}_5 \leq 0.5 \,\vert \text{entry})$ | 48.3 | 71.6 | 76.4 | 78.1 | 78.1 | 78.3 |
| 5. $\mathcal{P}(\text{Delay}_6 \leq 0.5 \,\vert \text{entry})$ | 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. $\mathcal{P}\,(\text{Delay} \le 0.5\,\vert\text{entry})$ (%) | 60.0 | 91.0 | 94.1 | 94.8 | 95.0 | 95.1 |
| 5. $\mathcal{P}\,(\text{Delay}_1 \le 0.5\,\vert\text{entry})$ | 60.3 | 90.7 | 94.0 | 94.7 | 94.9 | 94.9 |
| 5. $\mathcal{P}\,(\text{Delay}_2 \le 0.5\,\vert\text{entry})$ | 60.8 | 91.0 | 94.0 | 94.8 | 95.0 | 95.0 |
| 5. $\mathcal{P}\,(\text{Delay}_3 \le 0.5\,\vert\text{entry})$ | 60.6 | 91.0 | 94.1 | 94.9 | 95.1 | 95.1 |
| 5. $\mathcal{P}\,(\text{Delay}_4 \le 0.5\,\vert\text{entry})$ | 60.7 | 91.1 | 94.0 | 94.8 | 95.0 | 95.0 |
| 5. $\mathcal{P}\,(\text{Delay}_5 \le 0.5\,\vert\text{entry})$ | 59.6 | 91.1 | 94.4 | 94.9 | 95.3 | 95.3 |
| 5. $\mathcal{P}\,(\text{Delay}_6 \le 0.5\,\vert\text{entry})$ | 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. $\mathcal{P}\,(\text{Delay} \leq 0.5\,\vert\text{entry})$ (%) | 41.9 | 45.3 | 47.9 | 48.8 | 49.4 | 49.3 |
| 5. $\mathcal{P}\,(\text{Delay}_1 \leq 0.5\,\vert\text{entry})$ | 41.1 | 45.1 | 47.7 | 48.4 | 49.1 | 49.3 |
| 5. $\mathcal{P}\,(\text{Delay}_2 \leq 0.5\,\vert\text{entry})$ | 42.3 | 45.0 | 48.1 | 48.7 | 49.1 | 49.0 |
| 5. $\mathcal{P}\,(\text{Delay}_3 \leq 0.5\,\vert\text{entry})$ | 42.7 | 45.2 | 48.1 | 49.3 | 49.5 | 50.0 |
| 5. $\mathcal{P}\,(\text{Delay}_4 \leq 0.5\,\vert\text{entry})$ | 42.1 | 45.7 | 48.0 | 49.2 | 49.8 | 49.6 |
| 5. $\mathcal{P}\,(\text{Delay}_5 \leq 0.5\,\vert\text{entry})$ | 41.8 | 45.4 | 47.9 | 48.8 | 49.5 | 48.9 |
| 5. $\mathcal{P}\,(\text{Delay}_6 \leq 0.5\,\vert\text{entry})$ | 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.

|  | **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 4: 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 5: 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.

**Constraints** Our primary goal is to minimize $C$, the total staff required. A secondary goal, for given $C$, is to minimize $C+K$, the required number of trunk lines. We seek minimum values of $C$ and $K$ subject to the condition that constraints on performance are satisfied. We also aim to determine an appropriate agent-skill matrix $A$, subject to constraints on it and subject to constraints on performance. *The specific problem we consider here allows an arbitrary agent-skill matrix $A$ subject to the constraint that each agent have at most two skills.* We intend that to be illustrative of what can be done by our general approach.

We consider two performance constraints: First, we have a *speed-to-answer service-level target*; specifically, for each call type $i$, we require that

$$P(Delay_i \le \tau_i | entry) \ge \delta_i \ , \tag{4.1}$$

where $Delay_i$ denotes the steady-state delay for an arriving call of type $i$ (before beginning service) and in general the parameters $\tau_i$ and $\delta_i$ can depend on the call type $i$. The constraint in (4.1) is on the conditional delay probability given that the arriving call enters (is not blocked). Here we let

$$\tau_i = \tau = 0.5 \quad \text{and} \quad \delta_i = \delta = 0.80 \tag{4.2}$$

which corresponds to the requirement that 80% of the calls of each type be answered within 0.5 minute (30 seconds).

Our second performance constraint is a *blocking probability target*. For each call type $i$, we require that

$$P(Q^{(i)} = C + K) \le \epsilon_i \ , \tag{4.3}$$

where $Q^{(i)}$ is the steady-state total number of calls in the system seen by an arrival of type $i$. Under the assumption of Poisson arrivals (which we have assumed here, but which need not hold in practice), the random variables $Q^{(i)}$ 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

$$\epsilon_i = \epsilon = 0.005 \quad \text{for all} \quad i \ , \tag{4.4}$$

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.

Our approach is to exploit the resource-pooling property to obtain an initial candidate solution. Then we exploit simulation to make local adjustments to obtain a good solution.

**Finding an Initial Candidate Pair (C,K).** We make use of the resource-pooling property by initially acting as if all agents have all skills or, equivalently, as if there was only a single call type. Thus we choose initial values of $C$ and $K$ to meet the performance requirements in a standard $M/M/C/K$ model. Ways to do so, along with supporting theory, have been developed by Massey and Wallace (2004).

To apply the $M/M/C/K$ model, we need to calculate the aggregate parameters. The arrival rate for the $M/M/C/K$ model is the total arrival rate

$$\lambda \equiv \lambda_1 + \cdots + \lambda_n \tag{4.5}$$

and the expected service time is a convex combination of the individual mean service times, namely,

$$\frac{1}{\mu} \equiv \frac{1}{\lambda} \sum_{i=1}^{n} \frac{\lambda_i}{\mu_i} \ . \tag{4.6}$$

The ratio $\alpha \equiv \lambda/\mu$ is the *offered load* in the $M/M/C/K$ model. From heavy-traffic stochastic-process limits for many-server queues in Halfin and Whitt (1981), Puhalskii and Reiman (2000), Massey and Wallace (2004) and Whitt (2003), we know that, for the relatively large offered loads usually considered, that

$$C \approx \alpha + c_1\sqrt{\alpha} \quad \text{and} \quad K \approx c_2\sqrt{\alpha} \tag{4.7}$$

for suitable positive constants $c_1$ and $c_2$ (of order 1). Thus we can anticipate approximately what the critical values will be.

Since there are very fast algorithms for solving the $M/M/C/K$ model, it is easy to perform a search to find the best $(C, K)$ pair: A simple way is to first act as if $K = \infty$. With $K = \infty$, the speed-to-answer service-level is monotone in $C$, so it is easy to find the value of $C$ that just meets the speed-to-answer service-level target. Next, for that initial value of $C$ fixed, the blocking probability is monotone in $K$, so it is easy to find the value of $K$ such that the pair $(C, K)$ just meets the blocking probability target. However, since we decreased $K$, we increased the blocking, thereby reducing the delays for the customers that enter, so it may now be possible to decrease $C$. So, starting from the initial pair $(C, K)$, we can decease $C$ in unit steps, finding the associated $K$ to meet the blocking-probability target at each step, until we can decrease $C$ no more. Finding the new $K$ to go with $C - 1$ is relatively easy, because we can start with $K + 1$, where $K$ is associated with $C$. In that way, we can calculate the optimum $(C, K)$ pair for the $M/M/C/K$ model, i.e., the one yielding the minimum value of $C$, and for that $C$, the minimum value of $K$ such that both performance constraints are met.

The algorithm in Massey and Wallace (2004) based on the heavy-traffic limit can get close even more quickly.

We note that an infeasible value of $C$ can be identified if we locate an associated value of $K$ such that the two performance constraints are simultaneously violated. If we try to increase $K$ in order to reduce the blocking probability, the speed-to-answer service-level constraint will be violated even more. On the other hand, if we try to decrease $K$ to improve the speed-to-answer probability, then the blocking gets worse. So no value of $K$ can work with that value of $C$. (See Massey and Wallace (2004) for supporting theory.)

It is significant that this important initial step using the Erlang model can be generalized. By using the approximation algorithm in Whitt (2003), we can carry out the same procedure for the more general $M/GI/C/K{+}GI$ model, allowing abandonment with non-exponential service-time and customer-abandon-time distributions, but we do not do that here. For applications, that is an important extension, because statistical analysis of call-center data has shown that the service-time and abandon-time distributions are often not nearly exponential; see Brown et al. (2002).

**Determining the Work Group Sizes** Having used the $M/M/C/K$ model to find an initial pair $(C, K)$, we now must specify the skills and skill levels for the $C$ agents. We first determine the initial work group sizes; i.e., we first determine $C_i$, the number of agents in work group $i$, for each $i$, $1 \le i \le n$.

Applying the heavy-traffic stochastic-process limits for many-server queues again, we use a square-root approximation to allocate agents to the $n$ work groups. In particular, first ignoring integrality constraints, we first generate real numbers $R_i$ as initial estimates for $C_i$. We let

$$R_i = \alpha_i + x\sqrt{\alpha_i}, \quad 1 \le i \le n \ , \tag{4.8}$$

where

$$\alpha_i \equiv \frac{\lambda_i}{\mu_i} \tag{4.9}$$

is the offered load of call type $i$ and $x$ is a positive constant. It is easy to see that $x$ must be

$$x = \frac{(C-\alpha)}{\sum_{i=1}^{n} \sqrt{\alpha_i}} \ . \tag{4.10}$$

It is easy to see that, $R_i > 0$ and $R_1 + \cdots + R_n = C$, provided that $C > \alpha$, which will always be the case if the blocking probability is sufficiently small. As shown in Theorem 8.2.1 of Wallace (2004), the square-root method allocates relatively more capacity to the work groups with the smaller offered loads.

We next round the work group sizes $R_i$ specified in (4.8)–(4.10) to obtain the desired integer values $C_i$, keeping the property $C_1 + \cdots + C_n = C$. When there are several work groups, none of which are large, this rounding can cause difficulties, but these difficulties will be addressed in the second refinement phase of the algorithm, so we do not consider the rounding problem very important.

One simple way to do the rounding is to first round down, letting $C_i^* = \lfloor R_i \rfloor$, $1 \leq i \leq n$, where $\lfloor y \rfloor$ is the greater integer less than or equal to $y$. We start by assigning $C_i^*$ agents to work group $i$. The total number of agents assigned so far is

$$C^* \equiv C_1^* + \cdots + C_n^* \ . \tag{4.11}$$

We then need to assign the remaining $C - C^*$ agents to work groups. (Note that necessarily $0 \leq C - C^* \leq n - 1$.) A natural procedure is to assign the $C - C^*$ remaining agents one to each group in order of the differences $R_i - \lfloor R_i \rfloor$ (higher numbers first). A possible refinement designed to aid the smaller work groups is to allocate in order of the normalized values $(R_i - \lfloor R_i \rfloor)/R_i$ (again higher numbers first).

**Determining the Agent-Skill Matrix** Given the work group sizes, we next need to construct an associated initial agent skill matrix $A$. In actual applications, this step is likely to depend strongly on the skill combinations available. Assuming that resource pooling will be present, this step should not be too critical either. Nevertheless, we specify a specific procedure.

We start by noting that the first column of $A$ is determined so that $|G_k| = C_k$ for each $k$. Letting $C_{i,j,k}$ denote the number of agents in work group $i$ that have skill level $j$ to support call type $k$, we let

$$C_{i,j,k} = \begin{cases} C_i & \text{for } j = 1 \text{ and } k = i \\ \gamma_{i,j}(C - C_i)^{-1} C_k C_i, & \text{otherwise} \end{cases} \tag{4.12}$$

where $0 \leq \gamma_{i,j} \leq 1$ and it is understood that, for each pair $(i, k)$, we must have $C_{i,j,k} = 0$ for all but one value of $j$. The parameter $\gamma_{i,j}$ is the proportion of agents in work group $i$ that have a skill at level $j$.

In our examples each agent will have exactly two skills. When agents have two skills, $\gamma_{i,j} = 1$ for $j = 2$, and $\gamma_{i,j} = 0$ otherwise. Note that the scheme in (4.12) lets the number of agents in work group $i$ having secondary skill to serve call type $k$ be directly proportional to both $C_i$, the size of work group $i$, and $C_k$, the size of work group $k$.

Note that this framework allows us to consider partial skill assignments as well. For example, if we want only half of the agents in each work group to have secondary skills, then we let

$\gamma_{i,2} = 1/2$. Also note that, as before, we must do additional rounding to obtain integer values.

Given the above, it is not difficult to complete the definition of the agent-skill matrix $A$ in our setting, in which we only specify the number of skills each agent has. We do not discuss further details, because it does not seem too important and because we anticipate that there will be other constraints on the agent-skill matrix in applications.

**Using Simulation to Make Improvements.** Given the arrival and service parameters, and any agent-skill matrix $A$, we can use simulation to evaluate the performance. Thus we next use simulation to evaluate the performance and make refinements. We keep making modifications, trying to minimize the staff $C$, and minimize $K$ for given $C$, subject to the constraint that the performance requirements are met. In the initial phase above, we used resource pooling and fast numerical algorithms to get close to the optimal pair $(C, K)$. The goal is to be close enough that only a relatively small number of extra simulations are required to get to a good solution.

Since we have based our initial staffing on the resource-pooling property, we anticipate that the initial staffing requirement $C$ is optimistic (a lower bound). But to find out, we simulate the call-center model and examine the performance. If the speed-to-answer service-level target is not met for one or more classes, we increment $C$ by one and decrement $K$ by 1, thus keeping $C + K$ fixed, and repeat the experiment. By the process already specified, we determine new work groups and a new agent-skill matrix $A$. We repeat this step until all the speed-to-answer service-level targets are met. Because of the resource pooling, we anticipate that we will not need to increment $C$ many times.

Once we have found a staffing level $C$ meeting the speed-to-answer service-level target, we investigate the blocking-probability target. If the blocking-probability target is met, then we have an initial feasible solution, and we can go on to the refinement phase. If the blocking-probability target is not met, then we increment $K$ by one and conduct another simulation, continuing until the blocking-probability target is met. That might yield an infeasible $C$. If necessary, we increment $C$ by one and repeat the search for a feasible $K$, We thus complete this initial phase of the search with a feasible pair $(C, K)$ and a feasible agent-skill matrix $A$.

We do not stop, however, because it may be possible to do better. For example, the rounding to determine integer work-group sizes can cause difficulties. So we now enter a performance-based refinement phase. Our primary goal now is to decrease $C$. But now we use the observed performance to identify appropriate local changes to make. We first selectively decrement the total number of agents, C, by one and increment $K$ by one, thus again keeping $C+K$ fixed. The idea here is to remove an agent from the highest performing group, and then among those, one that has secondary skills serving another high performing group, where high-performing groups are those most strongly exceeding the speed-to-answer service-level target. We then update the agent-skill matrix $A$ by deleting the row and repeating the experiment. If all speed-to-answer service-level targets are met, then we try to repeat the procedure and delete another agent.

When no more agents can be simply removed, we try to make the last infeasible $(C, K)$ pair feasible by adjusting the agent skill mix. We switch skills of high performing work groups to low performing work groups. If by these switches we succeed in making the current $(C, K)$ pair feasible, then we go back to the previous step and try to simply decrement $C$. When the agent skill mix adjustment fails to produce feasibility, we go back to the last feasible $(C, K)$ and stop there.

The general idea is to perform a local search in the space of allowed pairs $(C, K)$ and skill matrices $A$, using simulations to evaluate the performance of each successive setting. Since performance requirements do not need to be generated in real time, the computer could be used to thoroughly investigate all possible local changes, if desired. The local search idea is similar to the local-search procedure used by Choudhury et al. (1995). Here, we can judge the progress of the refinements by comparing the number $C$ produced with the initial value produced in the resource-pooling step; that initial value should be a lower bound.

## 5. A Balanced Example

In this section we show how the provisioning algorithm in Section 4 works in a balanced example, closely related to the example used for the resource-pooling experiment in Section 3. Like Section 3, the balanced example has 6 call types. Again the mean service time is 10 minutes. The offered load for each call type is 13.75, so that the total offered load is 82.5.

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 6 exact analysis yields $(90, 20)$ instead, but the blocking probability is close to the boundary (0.5% blocking). Clearly, 89 agents are insufficient.

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 $(C, K) =$

|  | **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. $\mathcal{P}\left(\text{Delay} \le 0.5 \,|\text{entry}\right)$ (%) | 82.4 | 82.9 | 83.2 | 78.9 |
| 4. $\mathcal{P}\left(\text{Delay} \le 1.0 \,|\text{entry}\right)$ (%) | 89.6 | 90.0 | 90.5 | 87.0 |
| 6. Avg. Agent Util. (%) | 91.25 | 91.22 | 91.18 | 91.12 |

Table 6: 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.

$(108, 60)$. 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 just two skills.

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 7. We show the overall blocking probability, the overall mean delay, the mean delay for each call type, the overall speed-to-answer service-level $\mathcal{P}\left(\text{Delay} \le 0.5 \,|\text{entry}\right)$, the per-call-type speed-to-answer service-level $\mathcal{P}\left(\text{Delay}_i \le 0.5 \,|\text{entry}\right)$, 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.

From Table 7, 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%. Those speed-to-answer discrepancies could conceivably be due to statistical error, but the blocking gap is significant. Thus we conclude that we need to increment the number of agents, $C$.

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. For this symmetric example, we 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 7. 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 when each agent has all six skills.

## 6. 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

$$\alpha_1 = \alpha_2 = 4.25, \quad \alpha_3 = 10.50, \quad \alpha_4 = 13.75, \quad \alpha_5 = 19.25 \quad \text{and} \quad \alpha_6 = 30.50 \tag{6.1}$$

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

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 (6.1) are: $(7, 5)$, $(7, 5)$, $(14, 8)$, $(18, 9)$, $(24, 10)$, and $(36, 13)$, respectively, yielding a total requirement of $(106, 50)$. Interestingly, fewer resources are required in the single-skill case for the unbalanced model than for the balanced model, 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. In the initial simulation phase, where we simply increment $C$, the algorithm takes four steps, proceeding from $C = 90$ to $C = 93$. The successive work group sizes assigned when $C$ ranges from 90 to 94 are shown in Table 8. For this example, we rounded up the first five work-group sizes, and chose the final sixth work-group size so that the total number of agents became the correct number. The rounding procedure specified above would work better, but we will see that the rounding procedure is not critical.

Given the work-group sizes specified in the $C = 90$ column of Table 8, we next define the initial agent-skill matrix $A$. Following (4.12), we assign secondary skills to the agents as specified in Table 9. The initial agent-skill matrix itself is displayed in Appendix B.

| **Heuristic SBR Resource Provisioning Algorithm** | | |
|---|---|---|
| | **Number of Iterations (No. of Agents)** | |
| **Performance Measure** | **1 (90)** | **2 (91)** |
| 1. Blocking (%) | 0.54 | 0.43 |
| 2. Mean Delay (min) | 0.36 | 0.30 |
| 3. Mean Delay 1 | 0.37 | 0.32 |
| 3. Mean Delay 2 | 0.36 | 0.32 |
| 3. Mean Delay 3 | 0.35 | 0.30 |
| 3. Mean Delay 4 | 0.36 | 0.30 |
| 3. Mean Delay 5 | 0.35 | 0.28 |
| 3. Mean Delay 6 | 0.37 | 0.28 |
| 4. $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\vert \text{entry}\right)$ (%) | 79.8 | 82.7 |
| 5. $\mathcal{P}\left(\text{Delay}_1 \leq 0.5 \,\vert \text{entry}\right)$ | 79.5 | 81.9 |
| 5. $\mathcal{P}\left(\text{Delay}_2 \leq 0.5 \,\vert \text{entry}\right)$ | 79.7 | 81.9 |
| 5. $\mathcal{P}\left(\text{Delay}_3 \leq 0.5 \,\vert \text{entry}\right)$ | 80.0 | 82.5 |
| 5. $\mathcal{P}\left(\text{Delay}_4 \leq 0.5 \,\vert \text{entry}\right)$ | 80.0 | 82.6 |
| 5. $\mathcal{P}\left(\text{Delay}_5 \leq 0.5 \,\vert \text{entry}\right)$ | 80.3 | 83.5 |
| 5. $\mathcal{P}\left(\text{Delay}_6 \leq 0.5 \,\vert \text{entry}\right)$ | 79.7 | 83.9 |
| 6. Avg Agent Util (%) | 91.1 | 90.2 |
| 7. Work Group 1 Util | 91.2 | 90.4 |
| 7. Work Group 2 Util | 91.2 | 90.3 |
| 7. Work Group 3 Util | 91.1 | 90.4 |
| 7. Work Group 4 Util | 91.2 | 90.3 |
| 7. Work Group 5 Util | 91.2 | 90.4 |
| 7. Work Group 6 Util | 91.2 | 89.7 |
| 8. Work Group 1 Prim Util | 68.3 | 69.1 |
| 8. Work Group 2 Prim Util | 68.0 | 68.9 |
| 8. Work Group 3 Prim Util | 67.8 | 68.7 |
| 8. Work Group 4 Prim Util | 67.8 | 68.9 |
| 8. Work Group 5 Prim Util | 67.9 | 68.3 |
| 8. Work Group 6 Prim Util | 68.3 | 66.5 |

Table 7: 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 $\mathcal{P}\left(\text{Delay} \leq 0.5 \,\vert \text{entry}\right) \geq 0.80$.

| | | **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.** | $\alpha_i$ | **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 8: 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.

| **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 9: 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.

As indicated, the initial phase of the simulation algorithm in which we simply increment $C$ takes us to the feasible solution $(93, 18)$ in four steps. 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, 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 10. From Table 10, 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. 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 11. In the $8^{\text{th}}$ iteration (simulation run) we obtain a feasible solution with the pair $(91, 20)$, once again coming within a single agent of the single-group case in which all agents have 6 skills. The final 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 another $9^{\text{th}}$ 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%.

But we have admirably achieved our goal. Once again, the final feasible solution is within a single agent of what can be achieved when all agents have all skills.

| **First Phase of the SBR Resource-Provisioning Algorithm** | | | | |
|---|---|---|---|---|
| | **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. $\mathcal{P}\,(\text{Delay} \leq 0.5\,\vert\text{entry})$ (%) | 81.3 | 83.9 | 86.5 | 88.8 |
| 5. $\mathcal{P}\,(\text{Delay}_1 \leq 0.5\,\vert\text{entry})$ | 68.3 | 75.5 | 78.4 | 80.5 |
| 5. $\mathcal{P}\,(\text{Delay}_2 \leq 0.5\,\vert\text{entry})$ | 65.2 | 74.9 | 77.8 | 80.3 |
| 5. $\mathcal{P}\,(\text{Delay}_3 \leq 0.5\,\vert\text{entry})$ | 79.7 | 81.8 | 84.7 | 88.0 |
| 5. $\mathcal{P}\,(\text{Delay}_4 \leq 0.5\,\vert\text{entry})$ | 82.0 | 83.6 | 86.5 | 88.8 |
| 5. $\mathcal{P}\,(\text{Delay}_5 \leq 0.5\,\vert\text{entry})$ | 83.4 | 86.2 | 87.8 | 89.8 |
| 5. $\mathcal{P}\,(\text{Delay}_6 \leq 0.5\,\vert\text{entry})$ | 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 10: 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 $\mathcal{P}\,(\text{Delay} \leq 0.5\,\vert\text{entry}) \geq 0.80$.

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

Table 11: 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.

## 7. 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. The authors are grateful to the other co-advisors for their assistance. The second author was supported by National Science Foundation Grant DMS-02-2340.

| **Refined SBR Heuristic Resource Provisioning Algorithm** | | | | | | |
|---|---|---|---|---|---|---|
| | **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 12: 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 $\alpha_6 = 30.50$.

## References

Andradóttir, S., H. Ayhan and D. G. Down 2001. Server assignment policies for maximizing the steady-state throughput of finite queueing systems. *Management Science* **47**, 1421–1439.

Anton, J., V. Bapat and B. Hall 1999. *Call center performance enhancement using simulation and modeling.* Purdue University Press.

Azar, Y., A. Broder, A. Karlin and E. Upfal 1994. Balanced allocations. *Proceedings of the 26th ACM Symposium on the Theory of Computing (STOC)* 593–602.

Borst, S., A. Mandelbaum and M. I. Reiman 2004. Dimensioning Large Call Centers. *Operations Research*, **52**, 17–34.

Borst, S. and P. Seri. 2000. Robust Algorithms for Sharing Agents with Multiple Skills. Bell Labs, Lucent Technologies. Unpublished report.

Brigandi, A.,D. Dargon, M. Sheehan, and T. Spencer III 1994. AT&T's Call Processing Simulator (CAPS) Operational Design for Inbound Call Centers. *Interfaces* **24**, 6–28.

Brown, L., N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn, L. Zhao. 2002. Statistical analysis of a telephone call center: a queueing-science perspective. Department of Statistics, The Wharton School, University of Pennsylvania, Philadelphia, PA.

Choudhury, G. L., K. K. Leung and W. Whitt 1995. Efficiently providing multiple grades of service with protection against overloads in shared resources. *AT&T Technical Journal*, **74**, 50–63.

Gans, N., G. Koole, A. Mandelbaum. 2003. Telephone call centers: Tutorial, Review and Research Prospects. *Manufacturing and Service Opns. Mgmt.* **5**, 79–141.

Garnett, O. and A. Mandelbaum 2000. An introduction to skills-based routing and its operational complexities. Technion, Israel.

Garnett, O., A. Mandelbaum, M. I. Reiman. 2002. Designing a call center with impatient customers. *Manufacturing and Service Opns. Mgmt.*, **4**, 208–227.

Green, L. 1985. A Queueing System With General-Use And Limited-Use Servers. *Operation Research* **33**, 168–182.

Halfin, S., W. Whitt. 1981. Heavy-traffic limits for queues with many exponential servers. *Operations Research* **29**, 567–588.

Harrison, J. M. and M. Lopez 1999. Heavy Traffic Resource Pooling In Parallel-Server Systems. *Queueing Systems* **33**, 339–368.

Harrison, J. M. and A. Zeevi 2003. A method for staffing large call centers based on stochastic fluid models. Stanford University and Columbia University.

Hjlmtsson, G. and W. Whitt 1998. Periodic Load Balancing. *Queueing Systems* **30** 203–250.

Hopp, W. J. and M. P. Van Oyen 2003. Agile workforce evaluation: a framework for cross training and coordination. Northwestern University.

Jennings, O. B., A. Mandelbaum, W. A. Massey, and W. Whitt 1996. Server Staffing To Meet Time-Varying Demand. *Management Science* **42**, 1383–1394.

Koole, G. M. and J. Talim 2000. Exponential approximation of multi-skill call centers architecture. *Proceedings of QNETs 2000*, Ilkley, UK, **23**, 1–10.

Mandelbaum, A. and M. I. Reiman 1998. On pooling in queueing networks. *Management Science* **44**, 971–981.

Massey, W. A. and R. B. Wallace 2004. An optimal design of the $M/M/C/K$ queue for call centers. *Queueing Systems*, to appear.

Mitzenmacher, M. 1996. The Power Of Two Choices In Randomized Land Balancing. Ph.D. Dissertation, University of California at Berkeley.

Mitzenmacher, M. and B. Vöcking 1999. The asymptotics of selecting the shortest of two, improved. *Proceedings of the 1999 Allerton Conference on Communication, Control and Computing*, University of Illinois.

Perry, M. and A. Nilsson 1992. Performance modeling of automatic call distributors: assignable grade of service staffing. *Proceedings of the 14th International Switching Symposium*, 294–298.

Puhalskii, A. A., M. I. Reiman. 2000. The multiclass GI/PH/N queue in the Halfin-Whitt regime. *Adv. Appl. Prob.* 32, 564-595.

Ridley, A. 2003. Performance Analysis of a Multi-Class *Preemptive Priority Call Center with Time Varying Arrivals*, Ph.D. dissertation, University of Maryland, College Park.

Stanford, D. and W. K. Grassmann 1993. The Bilingual Server Systems: A Queueing Model Featuring Fully And Partially Qualified Servers. *Management Science* **31**, 221–277.

Stanford D. and W. K. Grassmann 2000. Bilingual server call centers. *Analysis of Communication Networks: call centers, traffic and performance*, eds. D. McDonald and S. R. E. Turner (*Amer. Math. Soc.*, Providence), 31–47.

Turner, S. 1996. *Resource Pooling In Stochastic Networks*, Ph.D Dissertation, Cambridge University.

Turner, S. (1998). The Effect of Increasing Routing Choice on Resource Pooling. *Probability in the Engineering and Informational Sciences.* **12**, 109–124.

Vvedenskaya, N. D., R. L. Dobrushin and F. I. Karpelovich 1996. Queueing systems with selection of the shortest of two queues: an asymptotic approach. *Problems in Information Transmission* **32**, 15–27.

Wallace, R. B. 2004. *Performance Modeling and Design of Call Centers with Skill-Based Routing*, Ph.D. dissertation, the George Washington University, School of Engineering and Applied Science.

Whitt, W. 1986. Deciding which queue to join: some counterexamples. *Operations Research* **34**, 55–62.

Whitt, W. 1992. Understanding the efficiency of multi-server service systems. *Management Science*, **38**, 708–723.

Whitt, W. 2002. Heavy-traffic limits for the $G/H_2^*/n/m$ queue. Department of Industrial Engineering and Operations Research, Columbia University.

Whitt, W. 2003. Engineering solution of a basic call-center model. Department of Industrial Engineering and Operations Research, Columbia University. Submitted to *Management Science.*

Williams, R. J. (2000) On dynamic scheduling of a parallel server system with complete resource pooling. *Analysis of Communication Networks: Call Centres, Traffic and Performance*, D. R. McDonald and S. R. E. Turner (eds.), Fields Institute Communications **28**, The American Math. Society, Providence, RI, 49–72.

Winston, W. 1977. Optimality of the shortest line discipline. *Journal of Applied Probability* **14**, 181–189.

Wolff, R. W. 1989. *Stochastic Modeling and the Theory of Queues*, Prentice Hall, Englewood Cliffs, NJ.

## 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 3.

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

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}^{(2)}_{90\times 6} = \begin{pmatrix} \mathbf{B}_{38\times 6} \\ \mathbf{C}_{52\times 6} \end{pmatrix}, \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 & 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 \\
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^{(2)}_{91\times 6}} = \begin{pmatrix} \mathbf{B_{40\times 6}} \\ \mathbf{C_{51\times 6}} \end{pmatrix}, \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 \\
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 \\
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 \\
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^{(2)}_{91\times 6}} = \begin{pmatrix} \mathbf{B_{42\times 6}} \\ \mathbf{C_{49\times 6}} \end{pmatrix}, \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}.$$