Establishing Agent Staffing Levels in Queueing Systems with Cross-trained and Specialized Agents

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Adindu Emelogu

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

The determination of the right number of servers in a multi-server queueing system is one of the most important problems in applied queueing theory. The problem becomes more complex in a system that consists of both cross-trained and specialized servers. Such queueing systems are readily found in the call centres (also called contact centres) of financial institutions, telemarketing companies and other organizations that provide services to customers in multiple languages. They are also found in computer network systems where some servers are dedicated and others are flexible enough to handle various clients' requests. Over-staffing of these systems causes increased labour costs for the underutilized pool of agents on duty, while under-staffing results in reduced revenue from lost customers and an increase in queue times. The efficient design and analysis of these systems helps management in making better staffing decisions. This thesis aims to develop models for establishing agent staffing levels in organizations with cross-trained and specialized staff with a view to minimizing cost and maintaining a desirable customer satisfaction. The work investigates the effect of various traffic loads on the number of agents required and the cost. It also considers how using specialized agents, flexible agents and a combination of both categories of agents affects the system. It uses a contact centre that has agents with monolingual, bilingual and trilingual (English, French and Spanish) capabilities to do the study.

Acknowledgements

Glory be to God whose love, mercy and goodness endure forever! I use this opportunity to thank my supervisor, Dr Winfried Grassmann, for his painstaking guidance, financial support and genuine encouragement in the course of this research. Without his help, I would not have been able to successfully produce this thesis.

I thank the members of my committee, Dr Eric Neufeld and Dr Grant Cheston, whose thoughtful insights and constructive comments on this thesis were indispensable. In the same vein, I thank the external examiner, Dr Keith Willoughby, for patiently reading my thesis and providing useful suggestions.

I would also like to appreciate the effort of the graduate correspondent, Ms Jan Thompson, who was always ready to give me useful pieces of advice. My thanks goes to all the faculty of the department for imparting a great knowledege to me, the staff members for providing selfless service and my fellow graduate students for making my experience in the department very memorable.

My special thanks is reserved for my dear family and friends. Their prayers, love and support were instrumental to the accomplishment of this research.

This thesis is dedicated to my lovely parents, Deacon and Mrs Stephen O. Emelogu.

Contents

List of Tables

List of Figures

List of Abbreviations

SAN	Stochastic Activity Networks
RAN	Recorded Announcement
TSF	Telephone Service Factor

Chapter 1

Introduction

Queueing systems arise readily in the industrial world whenever there is a congestion of traffic. Congestion may result from limited server resources and irregular client arrival patterns. There is the need to keep these systems orderly and efficient. Queueing theory has emerged as a well developed branch of applied probability theory that addresses issues in queueing systems.

The earliest systematic study of queueing systems can be credited to Danish mathematician Agner Krarup Erlang, the father of queueing theory. The publication of his pioneering work *The Theory of Probabilities and Telephone Conversation* in 1909 marked the beginning of this field of study [10]. Erlang's work in 1917 [4] contained formulae for loss and waiting time, which are now well known in the theory of telephone traffic. His work paved the way for the early application of queueing theory in the design and study of automatic telephone exchange in telecommunication industry. Subsequently, and as technological and mathematical methods developed, queueing theory began to find a wide range of applications in other areas such as manufacturing, marketing, military, health, government and computer networks. It is very likely that this interdisciplinary branch of knowledge which cuts across mathematics, statistics, operations research and computer science will continue to develop and be used to solve most of the queueing problems arising in client-server situations.

Some examples of queueing situations are people waiting to vote in an election, patients waiting to receive medical treatment in a hospital, cars waiting at booths in a toll plaza, programs waiting to be processed by a digital computer, customers waiting to place an order in the call center of a telemarketing company and packets waiting to be transmitted to a destination in a high-speed computer communication

network [7]. In all the cases, customers (people and jobs) are waiting to be served by servers or agents. The determination of the number of voting machines at elections, number of beds in a hospital, number of booths at a toll plaza to reduce traffic delays, number of processors to be made available in the computer and number of operators in a telephone system are some practical examples of agent-staffing problems often encountered in queueing systems.

In some of the systems, the customers may be heterogeneous, giving rise to more than one category of customers. Each category will require service from a server with the appropriate skill. There are situations where the servers are dedicated in which case they can only serve customers from a given category. In other situations, where some servers are cross-trained and flexible, they have the multi-skills required to serve customers from more than one category. In a call center, for example, agents can be trained to attend to customers in English, French or Spanish. Actually, it might not be feasible in real life situations to cross-train all agents due to cost, quality penalties, excessive stress and lack of flexible agents. The approach is therefore to have a mix of specialized, partially cross-trained and fully cross-trained servers. In this case, finding the optimal mix of these server categories to minimize cost and at the same time ensure that many customers do not balk or spend much time waiting in queues becomes very challenging. This thesis, therefore, aims to establish the staffing level in queueing systems that have specialized and flexible servers.

1.1 Agent Cross-training in Queueing Systems

Cross-training is a strategy used in queueing systems to achieve flexibility of servers in the face of variability. It involves equipping a server with more than one skill or combining two or more servers with different skills in the same department or team. In each case, the cross-trained server or the pooled team is able to give service to more than one customer type. Therefore, cross-training helps to increase the workforce agility in queueing systems. Cross-training servers helps to cope with variability issues in the queueing system. Variability can arise from customer arrival

rate fluctuations, service time variation, scheduling difficulties, waiting and balking characteristics. An increase in variability always results in a degradation of queueing system performance. Queueing systems with variability must be buffered by some combination of keeping customers on hold and building agent capacity. In human-based servers, flexible workforce capacity achieved from cross-training, skill-based routing and efficient scheduling are combined to reduce the effects of variability and improve the performance of the queueing system. The goal of cross-training is to reduce poor system performances like excessive congestion, poor service level, long waits and high abandonment. Cross-training also enables workers to develop a broad skill and experience base.

There are two basic components of cross-training architecture: the skill set design and the agent co-ordination or routing component. The skill set design determines how the agents are cross-trained. The agents could be specialized, overlapped (partially cross-trained) and fully cross-trained (fully flexible). In our study, we will combine the three categories.

The agent coordination component controls the scheduling mechanism and how customers are routed to the agents. This could be first-come-first-served (FCFS), queue/task priority, longest queue or longest wait scheduling principle. The right scheduling mechanism reduces capacity imbalance where some workers are over-utilized and some task-types are overstaffed. In our study, we will also combine these scheduling mechanisms appropriately.

In a queuing system where there are n categories of customers, each requiring a distinct type of service, there usually are various server types to meet the demands of the customers. If we assume that some servers are specialized, some partially cross-trained and others fully cross-trained, we will have a pool of servers where some are, for example, monolingual, bilingual or multi-lingual in capability.

Given n customer types, the maximum number of pools of i-lingual servers possible in the system is $\binom{n}{i}$ pools. The maximum number of pools of servers when n customer types are involved is given as $\sum_{i=1}^{n} \binom{n}{i}$ or $2^n - 1,\ n > 0$. For example, consider a system with 3-customer types in English, French and Spanish. We

can have the following pools of servers: English-servers, French-servers, Spanish-servers, English-French-servers, English-Spanish-servers, French-Spanish-servers and English-French-Spanish-servers. This means three pools of unilingual servers, three pools of bi-lingual servers and one pool of trilingual servers, giving a total of seven pools.

When the maximum number of pools of servers are used, a customer in a system with n categories of customers has a choice of $\sum_{i=0}^{n-1} \binom{n-1}{i}$ or 2^{n-1} server-types for service, $n > 0$. For example, consider an English customer and a system where there are maximum number of pools of servers in four languages, English, French, Spanish and Chinese. The customer can receive service by an English-server, English-French-server, English-Spanish-server, English-Chinese-server, English-French-Spanish-server, English-French-Chinese-server, English-Spanish-Chinese-server or English-French-Spanish-Chinese-server. This gives a total of 1+3+3+1 = 8 options.

It is noted that as the number of customer categories increases in a queueing system, the number of possible agent-pools that can be formed as well as the number of server-type options available for customer service increases geometrically. As a result, it may be difficult in practice, especially in human server systems, to find a situation where all the possible pools are employed for large categories of customers. For example, when a system consists of ten categories of customers, the number of server pools increases dramatically to 1023. Similarly, the number of server options available to a customer if all the pools are employed increases to 512.

1.2 Research Goals

The goal of this research is to seek a way of establishing the right staffing level in a queueing system with a combination of specialized, partially cross-trained and fully cross-trained agents in order to minimize cost. We propose to adapt it to a contact center where agents can attend to customers in three languages: English, French and Spanish. We will try to answer the following research questions in this work:

Is there any benefit to system performance of introducing cross-trained agents

when each of the customer categories has the same arrival rate? If yes, to what extent?

How do uneven traffic loads of the customer categories affect the staffing level in such queueing systems and what is the effect on cost?

1.3 Organization of Thesis

Chapter 1 gives the introduction to queueing systems and agent cross-training. Chapter 2 contains a literature review on contact centers, agent cross-training and finding the right number of servers. Chapter 3 describes our queueing models and optimization problems. It also contains a brief description of Möbius and Eqsp which are important tools that we will use in the study. It includes a simple sample queueing problem and how the tools are used to measure certain system performances. Our experiments are described in chapter 4, while chapter 5 gives the results of the experiments. We give the conclusion and possible areas of interest for future work in chapter 6.

Chapter 2

Literature Review of Contact Centres and Agent Cross-training

Several studies have been done to improve the design and management of queueing applications such as contact centres. A problem of particular interest is finding the right number of servers in queueing systems. Establishing an optimal staff level in a real-world queueing system makes such a system efficient and cost effective. Over-staffing results in an underutilization of the agents, and a resultant increase in the total cost of the system. On the other hand, under-staffing leads to long queues, long waits, loss of customers and the breakdown of agents. One practice of interest associated with queueing systems is cross training in which a staff member is equipped to serve more than one customer type. Agent cross-training helps to improve staffing and scheduling flexibility by making available many chances for matching agents to customers. This also reduces the total number of agents needed to handle a given call load. The design step in agent cross-training determines which agents are trained to handle which customer types, whereas the control step determines which customers are dynamically assigned to agents.

Mandelbaum [13] compiled a fairly complete list of academic publications on call centers. There are over 200 publications, arranged chronologically within subjects, each with its title, authors, source, full abstract and keywords.

In 1993, Stanford and Grassmann [19] presented a bilingual server system in a queueing model featuring fully and partially qualified servers. They slightly modified this work in [20] for a call centre providing service to a pool of customers with distinct service requirements: some simple, which can be rendered by all servers,

and some specialized (available only from cross-trained servers). In this study, the cross-trained servers are bilingual and so are able to serve customers in both the majority and minority language groups, while the specialized ones are monolingual and can only serve the majority language customers. The authors applied matrix geometric solutions to determine the minimum number of bilingual servers needed to achieve a satisfactory service for both language groups. Our work is similar to this because it can be applied to call centres. However, ours is an extension to trilingual server systems and uses economic optimization to derive results.

Andrews and Parsons [2] used economic optimization to establish the telephone-agent staffing levels in a telemarketing company. Their work provided an insight into some of the cost factors that are relevant in order-taking systems. However, they did not study a company with a mix of dedicated and cross-trained agents. Their model had an objective function with three expected-total-cost ingredients: (1) the cost of lost orders, (2) the cost of queueing time, and (3) the loaded cost of direct labour.

Andrews and Parsons found that an economically optimal level occurs at the point where the sum of all the three factors is at a minimum, provided that the service-level objective or the telephone service factor (TSF) is met. They used an expected-total-cost minimization algorithm to generate a variable-TSF staffing level as opposed to a fixed-TSF. They formulated the estimated economic impact of TSF service levels on the expected net profit from sales during each staffing period of thirty minutes as:

*the expected lost net profit from telephone orders during staffing period = (the expected number of calls per staffing period) * (the expected percent of calls generating orders) * (the average value of a permanently lost order) * (the expected percent of calls abandoned at the TSF service-level) * (the probability that a first abandonment is permanently lost).*

The abandonment rate is obtained by regressing the predicted variable, abandonment rate against the independent variable, TSF which transforms TSF percentages into corresponding abandonment rate percentages.

*The expected queuing cost during staffing period = (the expected number of calls per staffing period) * (the fraction of calls that went beyond a recorded announcement, RAN) * [(the expected queue time | queue time > RAN) - RAN] * (the average cost of connect time).*

*The expected labor cost during staffing period = (the number of agents on duty) * (the average loaded-wage per staffing period).*

The model assumes a given average work time for the operators during the half hour period to identify the optimum number of telephone agents required to cover a given number of calls expected during a half-hour staffing period. They applied their model and expected-total-cost minimization algorithm to L.L. Bean [2], a company that uses a total of about 900 part-time and full-time workers in its telemarketing operations, to generate a half-hourly economically optimal staff levels. They found that the overall estimated savings realized by using the variable-TSF approach instead of the fixed-TSF approach was considerable.

Bevilacqua Masi et al [3] analyzed the performance of a virtual call center consisting of a network of two stations. They defined two routing rules (external and internal) for this network, and used matrix-geometric techniques to derive the steady-state joint probability distribution for the number of customers at each station for each of the routing rules. The internal-rule configuration switches a customer to an alternate station after arrival at a primary station, based on server and other resource availability. The external-rule configuration switches a customer to an alternate station but only at arrival to the network contingent upon certain server availability conditions. The routing rules give us an idea of how to schedule our clients if the specialized servers are busy and there are two cross-trained server categories available for service.

The authors modeled the external routing rule system in such a way that an arriving customer joins a preferred queue among m single-server queues. However, the customer is re-routed to a shorter queue at arrival time if the number of customers, N_1, at the preferred queue is greater than or equal to some k, and the number of customers, N_2, at the alternate queue is less than k. In other words, if $N_1 > k$ and

$N_2 < k$ the call is re-routed but not if $N_1 > N_2 > k$. When $N_1 = k$ it becomes a shortest queue rule.

In the internal routing rule system, there are s connecting lines between the two stations allowing calls to be switched between the stations after arrival. A customer arrives directly to either station 1 or 2 depending on the number dialed. If all the servers in the original (primary) dialed station are busy, the automatic call distributor places the customer in the corresponding queue and questions both stations continuously for an idle server. The customer at the head of the queue is re-routed to the other station if a server becomes available at the other station and the line between the stations is available before a server becomes available at the original station. They assumed Poisson arrivals for the calls and exponentially distributed service times for the servers. They experimented for $s = k = 1$. They stated that the internal-rule system seems to utilize servers more efficiently than the external-rule system because of the longer time interval available for switching to the alternate station. In the case of the external-rule, since switching occurs only at arrival time, they concluded that there is a high probability of an idle server when a customer is queued. They established that the expected number of calls in external-rule systems is generally greater than that of internal-rule systems, and that for a multi-server system, sufficient connecting lines are needed in the internal-rule scheme to yield superior performance.

Chevalier et al [5] applied the theory of overflow analysis to derive an approximation for loss probabilities in a call center with specialized and cross-trained operators. Their model considered a call center serving N classes of calls, where N represents the number of different types of requests coming from customers. Each operator is able to answer one, several or all classes of calls. The model defined the following: the set C of all types of calls where $|C| = N$; the set M of operator pools where $m \in M$ is a pool of operators that can answer a set of C_m of calls with $C_m \subseteq C$; $p_m = |C_m|$ the number of different call types the operators of pool m can answer; S_m the number of operators in pool m and μ_m the mean service rate of operators in pool m. Calls are routed to the operators on a hierarchical flow basis in which

calls are first directed to specialized (1-polyvalent) pool of operators. If these servers are busy, the calls are routed to a 2-polyvalent, a 3-polyvalent pool of operators and so on until service is obtained. If all the consecutive servers are busy, the call is lost. They made the following assumptions for their model: each call type, c, arrives according to a Poisson distribution with parameter λ_c ; the arrivals of the different classes of calls are independent; the service time distribution is exponential with the mean service duration for operators in pool m given as $1/\mu_m$ (service rate only depends on the pool the call is handled and there is no service rate function of the call type); the call type is known before it enters the system, and lastly, calls do not wait for service (if a call finds all servers busy, it is lost and cleared from the system). Our model is very similar to the above model except that in our case, calls wait for service in a finite buffer until the buffer is full. It is when the buffer is full that an arriving call is lost.

They used simulation results from a call center that processes three types of calls to evaluate the quality of the proposed loss probability approximations. They found that when the arrival rates are high, the advantage of employing cross-trained operators decreases. In our study, besides the loss probability analysis, we will consider the cost to the system. We will try to find ways of mixing the staff in order to reduce the loss probability and optimize the total cost of the system.

Shumsky [17] looked into the approximation and analysis of a call center with flexible and specialized servers. Specifically, he presented a decomposition algorithm that estimates the performance of a call center that has only two server categories and two types of customers, A and B, each arriving according to a Poisson process with rates λ_A and λ_B respectively. There are N_A specialists who only serve type A customers, and N_F flexible severs who may serve either type. Regardless of the customer type, the service times are exponentially distributed and the mean service rates of specialist and flexible servers are μ_A and μ_F respectively. The queue discipline is such that type A customers prefer to visit A specialists, but will visit a flexible server if all specialists are busy. Flexible servers give non preemptive priority to type B customers. The model assumes that the service time distribution

depends on the servers and not on the customer type. The stability conditions are $\lambda_B < N_F\mu_F$ and $(\lambda_A + \lambda_B) < (N_A\mu_A + N_F\mu_F)$. The system is represented by a two-dimensional Markov process with states (X_1, X_2) where X_1 represents the total number of A-customers in queue or in service with an A-specialist, and X_2 represents the sum of B-customers and the number of customers of either type in service with a flexible server.

Instead of finding the steady state probabilities $P(X_1 = i, X_2 = j)$ directly, he divided the state space into four regions, two for each of the state space's two dimensions. The state space for the state variable X_1 is divided into two regions $(X_1 \leq N_A)$ and $(X_1 > N_A)$. For X_2, the state space is divided into regions $X_2 < N_F$ and $X_2 \geq N_F$. They claimed that this approach reduces the computation complexity. The following approximations form the heart of his algorithm:

$$P(X_1 = i|X_2 = j) \approx P(X_1 = i|X_2 < N_F), \quad X_2 < N_F \tag{2.1}$$

$$P(X_1 = i|X_2 = j) \approx P(X_1 = i|X_2 \geq N_F), \quad X_2 \geq N_F \tag{2.2}$$

$$P(X_2 = j|X_1 = i) \approx P(X_2 = j|X_1 \leq N_A), \quad X_1 \leq N_A \tag{2.3}$$

$$P(X_2 = j|X_1 = i) \approx P(X_2 = j|X_1 > N_A), \quad X_1 > N_A \tag{2.4}$$

He applied the model to a local utility's telephone call center where customers who can only see flexible servers are given priority for those servers. He compared his approximations with matrix geometric methods in terms of the average time in queue for the customers.

Tekin et al [21] studied the pooling strategies for call center agent cross-training. They focused mainly on the design step in agent cross-training which determines the agents that are trained to handle what customer types. Their model represented a departmental structure call center where the agents are divided into groups such that each customer is unambiguously assigned to a single department. The customer type to department assignment is one-to-one if the agent is dedicated (specialized) or many to one if the agent is cross-trained (flexible). They considered the impact of pooling, namely combining two or more departments into a larger department

with the agents in the pooled department cross-trained to handle all of the types of those departments. They investigated the effect of various system parameters such as arrival rates, mean service rates, variability in service times and the number of agents on the pooling decisions of how many departments to pool and which departments to pool. Since it might not be feasible in real life situation to cross-train all agents due to cost, quality penalties, excessive stress and lack of flexible agents, they recommended partial pooling scenarios. In partial pooling systems only some of the departments are pooled, while others continue to function as dedicated departments.

The authors specifically considered a call center with N dedicated departments and sought to pool $k \leq N$ departments into larger departments so as to minimize the average waiting time of customers in queue. In this case, the call center services N customer types and since the system has a departmental structure, customer type i is served by an agent in department $i, i = 1, 2, , N$. They assumed that customer type i arrives according to a Poisson process with mean λ_i, and requires a service time drawn from an independent identically-distributed sequence with mean T_i and squared coefficient of variation v_i^2. Department i has c_i servers. Hence, they modeled the initial system as N $M/G/c$ queues in parallel. The pooling of the queues defined by set K is the merging of the $k = |K|$ departments in set K by cross-training all servers in the pooled departments to handle all customer types in K. The interest was in determining which k departments to pool in order to achieve the largest reduction of average customer waiting time $W_i, i = 1, 2, ..., N$. They also assumed that the original system is stable, which requires $\rho_i = \lambda_i T_i / c_i < 1, i = 1, 2, ..., N$. They found that when mean service times differ greatly, pooling departments with a high ratio of mean service times may actually result in worse performance than not pooling, even if with very high utilization.

Grassmann [7] described useful methods of finding the right number of servers in a deterministic model, infinite server model and the equilibrium model. The so-called square-root formula for finding the optimal number of servers was originally suggested by Halfin and Whitt [11]. In the deterministic model and the infinite

server model, Grassmann randomized the model parameter $R = \lambda/\mu$ where λ is the arrival rate and μ is the service rate. This allowed him to deal with uncertain forecasts and queues in which the arrival rates vary over the day. In the deterministic setting, cost is minimized if the number of servers is set to the average traffic flow, $R = \lambda/\mu$, rounded to the next higher integer. In systems involving humans where the utilization of the human servers should not exceed a given percentage a, Grassmann established that the number of servers should be at least $100R/a$. Linder [12] observed that human servers cannot operate effectively at utilizations approaching 100 percent.

Chapter 3

Queueing Models and Optimization Problems

3.1 The Queueing Models

For the purpose of our study, we consider two queueing systems. The first model has two categories of customers and the second model has three categories of customers.

3.1.1 Two-category customer model

Our first queueing model considers a case where there are two categories of customers: customer type A and customer type B. The service agents are specialized or cross-trained. The specialized agents can attend to customers from only one category. Flexible agents can attend to customers from two categories. In other words, A-agents can serve only category A customers, B-agents can serve only category B customers, and A-B-agents can serve both type A and type B customers. Customers are served on a first-come, first-served basis. An arriving customer prefers to be served by an appropriate specialized server. If all the specialized servers are busy, the customer will be served by a flexible server. For example, a type A customer prefers to be served by an A-agent or an A-B agent in that order. When a flexible server becomes available, it chooses the longest-waiting customer from the category that has the longer queue. If the queues are equal at this time, it chooses from each queue with equal chance. Type A customers arrive at the rate of λ_A and type B customers arrive at the rate of λ_B. Service times depend on the server that a customer is routed to and not on the customer type that is being served. The

specialized agents serve customers at the rate of μ_1 and cross-trained agents serve at the rate of μ_2. All arrival times and service times are exponential. Furthermore, there is a queue of size Q_A for type A customers and a queue of size Q_B for type B customers. A customer that arrives when the queue is full balks and is lost.

An example of this queueing system is one where, for example, there are customers that want service in English and customers that want service in French. There are monolingual agents (specialized servers) who can speak either English only or French only, and bilingual agents (cross-trained servers) who can speak both languages. An arriving customer prefers to be served by a monolingual agent. If all the monolingual agents that can serve him are busy, he is served by a bilingual agent. If all the bilingual and monolingual servers are busy and the queue for their language is not full, he waits in the queue. If the queue is full, he is lost. We investigate the combination of specialized and cross-trained agents that will result in the optimal performance of the system. In other words, we assess the effects (costs and benefits), if any, of cross-trained agents on system performance measures.

3.1.2 Three-category customer model

In our second model, we introduce an extra category of customers into the system. For example, customers that want service in Spanish. We investigate if there is any significant improvement on system performance of introducing trilingual agents. Here, we consider a queueing system where there are three categories of customers: customer type A, customer type B and customer type C. The service agents are specialized, partially flexible (partially cross-trained) or fully flexible (fully cross-trained). The specialized agents can attend to customers from only one category. Partially cross-trained agents can serve customers from only two categories, and fully flexible agents can attend to customers from all three categories. In other words, A-agents can serve only category A customers, B-agents can serve only category B customers, C-agents can serve only category C customers, A-B-agents can serve both type A and type B customers, A-C-agents can serve both type A and type C customers, B-C-agents can serve both type B and type C customers, and A-B-C-

agents can serve all three customer types. Customers are served on a first-come, first-served basis. An arriving customer prefers to be served by an appropriate specialized server. If the servers are busy, the customer will be served by a partially flexible server. If these two server categories are busy, the customer is served by a fully flexible server. For example, a type-A customer prefers to be served by an A-agent, an A-B/ A-C agent or an A-B-C agent in that order. When a partially flexible server becomes available, it chooses the longest-waiting customer from the category that has the longer queue. If the two queues are equal at this time, it chooses from each of the queue with equal chance. When a fully cross-trained server becomes available, it chooses the longest-waiting customer from the category that has the longest queue. If all the queues are equal, it chooses from each queue with equal chance. Type-A customers arrive at the rate of λ_A, type B customers arrive at the rate of λ_B and type C customers arrive at the rate of λ_C. The specialized agents serve customers at the rate of μ_1, partially cross-trained agents serve at the rate of μ_2 and fully cross-trained agents serve at the rate of μ_3. Service times are server-dependent and do not depend on the type of customer that is being served. All arrival times and service times are exponential. Furthermore, there is a Q_A finite queue size for type-A customers, Q_B queue size for type-B customers and Q_C queue size for type-C customers. Any customer that arrives when the queue is full balks and is lost.

3.2 Motivations

The need to cope with variation issues and obtain satisfactory performance measures in queueing systems has necessitated an increase in interest in the efficient design and management of these systems. Contact centers proliferate in the global economy, and are among the various queueing systems that consist of specialized and cross-trained operators. Agent cross-training is a practice that is usually employed to improve staffing and scheduling flexibility by making available more chances for matching agents to customers, and reducing the total number of agents needed to handle a

given call load. The design step in agent cross-training determines which agents are trained to handle which customer types, whereas the control step determines which customers are dynamically assigned to agents.

A performance analysis tool is important in comparing the performance of various system configurations, and it helps management in making staffing optimization decisions. The performance of a contact center, for example, can be measured in terms of several metrics such as the mean waiting time, the mean service time, the loss probability, and the joint distribution of each customer type in the system. Our model is simple enough to enable us to compute these performance measures. Besides, theoretical tools and software exist to help us in the analysis of our model. Furthermore, it is not difficult to modify the model to represent other systems. For example, it can be applied to computer network systems where some servers are dedicated and others are flexible to handle various clients' requests.

3.3 The Optimization Problem

The basic form of optimization is to identify the alternative ways of achieving a given objective and then to select the alternative that accomplishes the objective in the most efficient manner, subject to constraints on the ways. The problem is to optimize the value of an objective function, subject to any resource and/or other constraints such as legal, input, environmental, and behavioral restrictions.

Basic economic decision analysis involves determining the action that best achieves a desired goal or objective. This means finding the action that optimizes the value of an objective function. For example, in a production problem, one may want to find the combination of resources that minimizes cost. In a price-output decision-making problem, the goal may be to determine the output level that maximizes profits. In the case of our study, the optimization problem involves finding the combination of agent categories that minimizes the cost of a queueing center while satisfying some system performance requirement constraints. These constraints include not exceeding a given level of client loss, line length and mean waiting time.

Mathematically, we can represent an optimization problem as:

Optimize

$$y = f(x_1, x_2, ..., x_n) \tag{3.1}$$

subject to

$$g_j(x_1, x_2, ..., x_n) \begin{pmatrix} \leq \\ = \\ \geq \end{pmatrix} b_j \quad j = 1, 2, ..., m \tag{3.2}$$

where equation 3.1 is the objective function and equation 3.2 constitutes the set of constraints imposed on the solution. The x_i variables, $x_1, x_2, ..., x_n$, represent the set of decision variables, and $y = f(x_1, x_2, ..., x_n)$ is the objective function expressed in terms of these decision variables. As indicated in equation 3.2, each constraint can take the form of an equality ($=$) or an inequality ($\leq$ or $\geq$) relationship. Depending on the nature of the problem, the term *optimize* means either *maximize* or *minimize* the value of the objective function. The task of maximization and minimization are trivially related to each other, since one person's function f could just as well be another person's *-f*.

There are various techniques for solving optimization problems. Some of them include:

- Differential calculus
- Search methods
- Lagrange multipliers method
- Mathematical programming methods

Each method has the class of optimization problem to which it is best suited. The simplest situation is the unconstrained optimization problem. In an unconstrained optimization problem, no constraints are imposed on the decision variables and so

there is no equation 3.2 attached to the optimization problem. Two classes of problems can be identified here [16], one in which the information about the derivative of the function to be optimized is known and one in which the derivative is not known. For the latter, the Golden Section Search Method and the Successive Parabolic Interpolation Method are used when the function is a one-variable function [14]. The Nelder-Mead (also called Downhill Simplex Method) is used for multidimensional objective functions.

When the derivative is known, the Newton's method, Steepest Descent (also called the Gradient Search Method) and the Conjugate Gradient Search method can be used.

Another form of optimization problem is one in which all the constraints of the problem can be expressed as equality (=) relationships. In this case, it can be shown that the optimal point must lie on the boundary of the feasible region. According to Foulds [6], the Lagrangian multipliers method and the Jacobian method can be used to solve this type of problem.

When the constraints in an optimization problem take the form of inequality relationships ($\leq$ or $\geq$) rather than equalities, as is often the case, mathematical programming techniques are used to solve such class of problems. Mathematical programming techniques include:

Linear Programming: In a linear programming problem, both the objective function and the constraint relationships are expressed as linear functions of the decision variables.

Integer Programming: Here, some or all of the decision variables must assume integer values.

Quadratic Programming: In quadratic programming problems, the objective function or the set of constraints is expressed as a quadratic function of the decision variables.

Algorithms are available for solving optimization problems that meet these requirements.

There are some factors that can make optimization problems fairly complex and

difficult to solve, and sometimes render them unsolvable by formal optimization procedures.

One such complicating factor is the existence of multiple decision variables in a problem. An optimization problem that has just two decision variables can easily be solved by graphing the constraints. As the number of decision variables increases, the dimensionality and the complexity of analysis increase. In manufacturing industries, relatively simple procedures exist for determining the profit-maximizing output level for the single-product firm. However, the typical medium-size or large-size firm often produces a large number of different products, and as a result, the profit-maximization problem for such a firm requires a series of output decisions. This means one output decision for each product.

Another factor that may add to the difficulty of solving a problem is the complex nature of the relationships between the decision variables and the associated outcome. For example, in public policy decisions on government spending for such items as education, it is extremely difficult to determine the relationship between a given expenditure and the benefits of increased income, employment, and productivity it provides. No simple relationship exists among the variables. In our study, we need to keep the objective function relatively simple and ensure that only situations where a relationship could be established between the decision variables and the outcome variable are considered.

A third complicating factor is the possible existence of one or more complex constraints on the decision variables. For example, virtually every organization has constraints imposed on its decision variables by the limited resources—such as capital, personnel, and facilities—over which it has control. In our study, there is a finite buffer size and there is also a limit to the number of agents that could be hired, and at the same time we want to achieve desirable system performance with these constraints. Such constraints must be incorporated into the decision problem. Otherwise, the optimization techniques that are applied to the problem may yield a solution that is not feasible and is therefore unacceptable from a practical standpoint.

Another complicating factor is the presence of uncertainty or risk. Analyzing

decision making problems when the outcome of each action is known with certainty is simpler than analyzing decisions involving risk and uncertainty.

3.4 The Optimization Model

For simplicity, we consider the following cost ingredients in the system:
S_L, cost due to the loss of customers when the queue is full
S_W, cost due to customers waiting in the queue when all the servers are busy
S_1, cost of a specialized agent per time unit
S_2, cost of a partially cross-trained agent per time unit
S_3, cost of a fully cross-trained agent per time unit
We also define S_G, cost of employing agents over the work period as

$$S_G = S_1 n_1 + S_2 n_2 + S_3 n_3 \tag{3.3}$$

where n_i is the number of agents used in each category of agents. Therefore, the total system cost to be optimized is given as $T_C = S_L + S_W + S_G$.

We make the reasonable assumption that $S_1 \leq S_2 \leq S_3$. This assumption is true in real life because it is easier, for example, to find agents that can speak only one of English, French or Spanish than those that can speak all three. We also assume that the cost of all specialized agents are equal; that the cost of all partially cross-trained agents are equal and that the cost of all fully cross-trained agents are equal. In order to demonstrate this principle in our study, we assume the sample costs $S_1 = 10$ dollars, 22 dollars for the cost per lost customer and 3 dollars for the cost of a unit time waited in the queue by a customer.

A similar reasoning is true for the service rate of the agents. In this case, let:
Service rate of a specialized agents, μ_1
Service rate of a partially cross-trained agent, μ_2
Service rate of a fully flexible agent, μ_3
Then, $\mu_3 \leq \mu_2 \leq \mu_1$. This assumption is likely to be true because due to constant practice and specialization, a dedicated agent is likely to be faster than a flexible

agent. We also assume that the expected service times of all specialized agents are equal; that the expected service time of all partially cross-trained agents are equal and that the expected service time of all fully cross-trained agents are equal.

3.5 Solution Methods and Description of Tools

We give an introduction to the methods of solution that we will use in solving the queueing problems. We also describe the basic tools of Möbius [15] and the Eqsp package [9], which we will use in obtaining and analyzing solutions. The optimization problem will be solved by applying cost minimization techniques using simple search methods to select the best agent mix.

Queueing problems in particular and stochastic problems in general can be analyzed by analytic methods, numerical methods and Monte Carlo simulations [8]. Analytical methods use explicit formulas to express the relationship between the variables of a model. Simple formulas are ideal but are usually not available for many classes of queueing problems. When analytical methods are available, they yield an exact solution to a problem. Complex formulas that form the basis of an algorithm may be beneficial especially if they yield simple results. However, they may be very hard to deal with if many variables are involved.

In numerical methods, one does not need to know the underlying mathematical theories and details involved in a complex algorithm before performing experiments. One fixes all the input parameters of the model to certain values and calculates probabilities, appropriate distributions and expectations of interest. The iterative method is an approach that can be applied in numerical methods. In numerical computational mathematics, an iterative method attempts to solve a problem (for example, finding the root of an equation or system of equations) by finding successive approximations to the solution starting from an initial guess and improving the solution. However, some analytical results obtained from numerical methods could be inefficient when the desired solution requires a long time to converge. The solutions obtained are only accurate to a degree of tolerance specified. In obtaining the

stationary probability vector from either the stochastic transition probability matrix or from the infinitesimal generator, the only operations in which the matrices are involved are multiplications with one or more vectors, or with preconditioners. These operations do not alter the form of the matrix, and result in compact storage schemes and low memory requirement. This approach is in contrast to direct methods, which attempt to solve the problem by a finite sequence of operations, and, in the absence of rounding errors, would deliver an exact solution (like solving a linear system of equations $Ax = b$ by Gaussian elimination). However, the elimination of nonzero elements of the matrix during the reduction phase often results in the creation of several nonzero elements in entries that previously contained zeros. The resulting fill-in makes compact storage difficult and may require exhaustive memory usage.

Iterative methods are usually the only choice for nonlinear equations. Iterative methods are often useful even for linear problems involving a large number of variables (sometimes of the order of millions), where direct methods would be prohibitively expensive (and in some cases impossible) even with the best available computing power. The direct method for solving Markov chains has a computational complexity of the order $O(n^3)$, where n is the number of states. The iterative method has an order of complexity of about $O(n^2)$, which makes a huge difference in computational speed when large problems are involved. The sparsity of the matrices generated by such large problems which often occur in real life makes the iterative methods faster and more preferable than the direct method. For this reason Möbius [15] which uses the iterative method is more efficient than Eqsp in multi-dimensional problems.

In simulation, one randomly selects a large number of outcomes, and for each outcome, evaluates the system response. Probabilities and other measures of interest are estimated by averaging all selected outcomes. For stochastic problems that are mathematically intractable and where there is no known mathematical formula or underlying theory, simulation can be the last resort. Even for problems that are mathematically tractable but whose solution may be cumbersome and time-consuming, simulation can often provide a higher level of detail than can other

techniques. Simulation is the preferred method of conducting experiments which in real life would be very risky and costly in physical and political respects. The computational time complexity in simulation increases only linearly with the number of states, making it more efficient than the numerical method when analyzing large and complex stochastic systems.

However, it is difficult to develop cause-and-effect relationships through simulation, especially when the system under consideration requires the specification of many input parameters and involves complex interactions. The statistical analysis of simulation results is difficult because many questions are involved. What is the effect of the starting conditions of the simulation on the final results? How many data points should be disregarded as reflecting primarily the starting conditions and not the long-term characteristics of the simulated system? In the course of a parametric analysis, have we discovered a local or a global optimum set of operating conditions? What is the statistical confidence that can be attached to the results? To reduce the confidence interval by a factor k, one needs to increase the computational effort by a factor of k^2. Like all empirical techniques, it is easy to underestimate the resources needed to develop, validate and run a simulation model. It is therefore expensive to obtain very accurate results by simulation.

The models present us with a queueing problem and optimization problem. The queueing problem will be solved using solutions to Markov chain models. The solution to the queueing problem will yield such performance measures like queue length, line length, loss probability and server utilization when various combination of agents are used. M*ö*bius is the available software tools that we will use. We use Eqsp to confirm the results obtained in M*ö*bius. Basically, each of these tools employs analytical methods based on continuous time Markov chains (CTMC) to obtain probability solutions. For equilibrium solutions, M*ö*bius uses the steady-state iteration method whereas Eqsp uses the direct method. For transient solutions they use the randomization method (also called uniformization method).

3.5.1 Möbius

Möbius is a software tool developed by William Sanders and his PERFORM team at the University of Illinois at Urbana-Champaign. Möbius is currently being applied by a wide range of users to study the performance and dependability of systems. It is one of the most important tools that we will be using in our study. Möbius uses the stochastic activity network (SAN) approach which is a stochastic extension to Petri nets, otherwise known as the generalized stochastic Petri net. Invented in 1962 by Carl Adam Petri, a Petri net is a non-timed, formal, graphical, executable, mathematical modeling tool used for the specification and analysis of concurrent, discrete-event dynamic systems. It consists of places, transitions and arcs that connect them. Input arcs connect places with transitions, while output arcs start at a transition and end at a place.

Using graphical primitives, stochastic activity networks provide a high-level modeling formalism with which one can specify the performance and dependability of a system. SANs consist of four primitive objects: activities, places, input gates, and output gates .

Activities represent actions of the modeled system. There are two types of activities: timed and instantaneous. Timed activities have durations that impact the performance of the modeled system, such as the service time of a customer, a packet transmission time or the time associated with a retransmission timer. Timed activities are represented graphically as thick vertical lines. Each timed activity has an activity time distribution function associated with its duration. Activity time distribution functions can be generally distributed random variables. Each distribution can depend on the marking of the network. For example, one distribution parameter could be a constant multiplied by the marking of a certain place. Instantaneous activities represent actions that complete immediately when enabled in the system. They are represented graphically as thin vertical lines.

Places represent the state of the modeled system. Each place is represented graphically as a circle that contains a certain number of tokens, which represents

the marking of the place. The set of all place markings represents the marking of the stochastic activity network. The tokens in a place are homogeneous, and as such, only the number of tokens in a place is known. Consequently, there is no identification of different kinds of tokens within a place. The meaning of the marking of a place is arbitrary. For example, the number of tokens in a place could represent a number of objects, such as a number of tasks awaiting service. Alternately, the number of tokens in a place could represent an object of a certain type, such as a task with a certain priority level. This dual nature of a place marking provides a great deal of flexibility in modeling the dynamics of a system.

Case probabilities, represented graphically as circles on the right side of an activity, model uncertainty associated with the completion of an activity. Each case stands for a possible outcome, such as a routing choice in a network, or a failure mode in a faulty system. Each activity has a probability distribution, called the case distribution, associated with its cases. This distribution can depend on the marking of the network at the moment of completion of an activity. If no circles are shown on an activity, one case is assumed to have a probability of one.

Input gates control the enabling of activities and define the marking changes that will occur when an activity completes. They are represented graphically as triangles. On one side of the triangle is a set of arcs to the places upon which the gate depends, also called input places. Each input gate is defined with an enabling predicate and a function. The enabling predicate is a Boolean function that controls the enabling of the connected activity. It can be any function of the markings of the input places. The default scenario of the input gate is when a place is directly connected to an activity with an arc. This situation is equivalent to an input gate with a predicate that enables the activity whenever the place has more than zero tokens along with a function that decrements the marking of the place whenever the activity fires. When an activity fires it means that it has taken place.

Output gates define the marking changes that will occur when activities complete. Unlike the input gate, the output gate is only associated with a single case. An output gate is represented graphically as a triangle with its flat side connected to an activity

or a case. On the other side of the triangle is a set of arcs to the places affected by the marking changes. An output gate is defined only with a function. The function defines the marking changes that occur when the activity completes.

There is also a default scenario for output gates. If an activity is directly connected to a place, it is equivalent to an activation in which an output gate has a function that increments the marking of the place whenever the activity is fired.

3.5.2 A Sample Queueing Model Solution in Möbius

In Möbius, the basic building blocks of a large and complex model are the atomic models. The atomic sub-models are combined in order to construct the whole model. After combining the sub-models, the next step is to define a set of measures of interest on the model. Finally, the values of these measures are computed using a selected solution method. One can also investigate how these values are affected by a change in the model parameters. In this sub-section, we use Möbius to model a simple M/M/c/K queueing problem. While we go through the solution, we will describe the essential features, models and frameworks of Möbius, including:

- Atomic Models
- Composed Models
- Reward Models
- Study Models
- State space Generation Models
- Solution techniques

An M/M/c/K queue is a birth-death multi-server model in which customer arrivals are Poisson, the service times are exponentially distributed and there is a finite buffer size. Here, K stands for the maximum number of customers allowed in the system. Therefore, $K - c$ is the buffer size.

We start by building the atomic model. Möbius supports the use of SAN formalism to build atomic models.

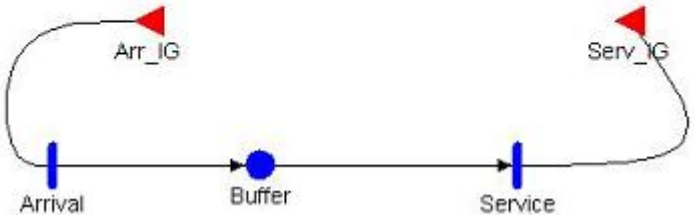


Figure 3.1: An M/M/c/K Atomic Model

Figure 3.1 is the graph of the atomic model. It consists of an input gate and an output gate represented with triangles, a place (buffer) represented by a circle, two thick vertical lines representing timed-activities (arrival and service) and interconnecting lines .

Two or more atomic models can be combined to form a composed model. A composed model links sub-models together and allows them to interact when they have a shared state variable. Our simple example has only one atomic model and so a composed model is not necessary.

Reward models are used to specify performance measures on atomic and composed models. Möbius implements a performance variable (PV) type of reward model. A performance variable can be specified to be measured at an instant of time, in steady state, be accumulated over a period of time, or be time-averaged over a period of time. In our example, we consider steady state solutions. Our reward performance measures of interest could be the occupancy of the system, the utilization, the average response time and the loss probability. Once the rate and impulse rewards are defined, we can specify the desired statistics on the measure. The options include solving for the mean, variance, or distribution of the measure, or for the probability that the measure will fall within a specified range.

In the study model, we assign values to the variables. During the specification of atomic, composed, and reward models in Möbius, global variables can be used to

parameterize model characteristics. A global variable is a variable that is used in one or more models, but not given a specific value. Models are solved after each global variable is assigned a specific value. In this example, the global variables include $ArrivalRate$, $ServiceRate$, num_server and BUF_SZ. One such assignment forms an experiment. Experiments can be grouped together to form a study. In one experiment, we let the maximum number that can be in the system, $K = 5$, an arrival rate of 4.0 per hour, a service rate of 3.0 per hour and a server number of 3.

Two classes of solution techniques exist in Möbius: discrete event simulation and state-based, analytical techniques. The analytical options include transient solutions and steady-state solutions. Any model specified using Möbius may be solved using simulation. Only the models that have exponentially distributed delays, or have no more than one concurrently enabled deterministic delay can be solved using analytic techniques applied to a generated state space. We prefer analytical solutions to simulations because analytical solutions give more accurate results of the performance measures.

The first step in analytic solution with the Möbius tool is the generation of a state space, done by the state-space generator. The state-space generator may be employed on any Möbius model. This allows the state-space generator to be generic, so it need not understand the semantics of a model on which it is operating. Once the state space is generated, an analytical method is then employed to solve for the required performance variables. In our example, $K + 1 = 6$ states are generated. This means that there could be 0, 1, 2, 3, 4 or 5 customers in the system at any point in time.

The model is finally solved using the iterative steady state solver to generate results for the performance variables. Refer to the Appendix for the documentation of the solution of this model.

Möbius supports two modes of discrete event simulation: transient and steady state. In the transient mode, the simulator uses the independent replication technique to obtain statistical information about the specified reward variables. In the steady-state mode, the simulator uses batch means with deletion of an initial tran-

Table 3.1: MES Description of an M/M/c/K Queue

Event	**Effect On** X1	**Rate**	**Condition**
Arrival	+1	λ	X1$\leq K-1$
Service1	-1	$\mu \cdot X1$	X1$\leq c-1$
Service2	-1	$\mu \cdot c$	X1$\geq c$

sient to solve for steady-state, instant-of-time variables. Estimates available during simulation include mean, variance, interval, and distributions. Confidence intervals are computed for all estimates. The simulator may be executed on a single workstation, or distributed on a network of workstations.

3.5.3 Queueing Model Solution in Eqsp

We can as well solve the above queueing model example in Eqsp, a program developed by Grassmann. We check if we can get the same results in Eqsp as in Möbius. We start by describing the model as a Markovian Event System (MES). This shows us an event, its effect on the system variables, the conditions for which the event occurs and the rate of occurrence. We notice that more events are needed to describe the system than are needed in Möbius. This arises from the fact that an activity in Möbius that can occur from more than one condition is split into a number of events in Eqsp equal to the number of conditions.

Table 3.1 shows the description of the M/M/c/K queueing model. The system has only one variable, X1 which represents the number of customers in the system. We set an initial restriction to this variable such that it cannot be less than zero. There are three events associated with the system. Arrival, Service1 and Service2. Arrival increases the state variable X1 by 1 at rate λ. The other events, Service1 and Service2 decrease X1 by 1. However, the rate at which they occur depends on the state value of X1. In Möbius, these two events are combined into one and the rates are specified in the rate function environment. We solved the model with

the parameters $\lambda = 4$, $\mu = 3$, $c = 3$ and $K = 5$. This generated 6 states and 10 transitions. The measures we calculated were the expected value of X1, the joint distribution and the marginal distributions. The results match the ones we obtained when we used Möbius. $Exp(X1) = 1.39213, Prob(X1 = 0) = 0.2583376, Prob(X1 = 1) = 0.3444501, Prob(X1 = 2) = 0.2296334, Prob(X1 = 3) = 0.1020593, Prob(X1 = 4) = 0.0453597, Prob(X1 = 5) = 0.0201599$. However, it is not advisable to use the Eqsp package in problems that generate large state spaces because for such problems it implements a method that is slower than the one used by Möbius.

3.5.4 Queueing Model Analytical Solution

Mathematical formulas for calculating the steady state probabilities and other measures of interest in M/M/c/K queues have been derived in many operations research textbooks and literature that discuss queueing models, such as in [1] and [18].

The state transition diagram of an M/M/c/K queue is given in Figure 3.2.

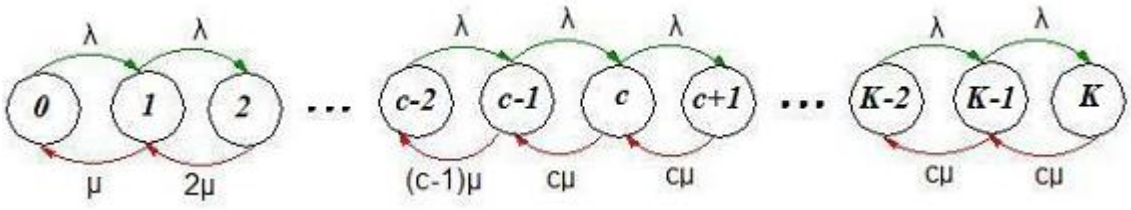


Figure 3.2: State Transition Diagram of an M/M/c/K Queue

In a birth-date process, the steady sate probabilities are obtained by solving the iteration:

$$\begin{cases} p_{n+1} = \frac{\lambda_n + \mu_n}{\mu_{n+1}} p_n - \frac{\lambda_{n-1}}{\mu_{n+1}} p_{n-1} \quad (n \geq 1) \\ \\ p_1 = \frac{\lambda_0}{\mu_1} p_0 \end{cases} \tag{3.4}$$

and

$$p_2 = \frac{\lambda_1 + \mu_1}{\mu_2} p_1 - \frac{\lambda_0}{\mu_2} p_0 \tag{3.5}$$

$$= \frac{\lambda_1 + \mu_1}{\mu_2}\frac{\lambda_0}{\mu_1}p_0 - \frac{\lambda_0}{\mu_2}p_0$$

$$= \frac{\lambda_1\lambda_0}{\mu_2\mu_1}p_0 \tag{3.6}$$

$$p_3 = \frac{\lambda_2 + \mu_2}{\mu_3}p_2 - \frac{\lambda_1}{\mu_3}p_1 \tag{3.7}$$

$$= \frac{\lambda_2 + \mu_2}{\mu_3}\frac{\lambda_1\lambda_0}{\mu_2\mu_1}p_0 - \frac{\lambda_1}{\mu_3}\frac{\lambda_0}{\mu_1}p_0$$

$$= \frac{\lambda_2\lambda_1\lambda_0}{\mu_3\mu_2\mu_1}p_0 \tag{3.8}$$

and so on.

We can see that the emerging pattern is thus:

$$p_n = \frac{\lambda_{n-1}\lambda_{n-2}...\lambda_0}{\mu_n\mu_{n-1}...\mu_1}p_0, (n \geq 1) \tag{3.9}$$

$$= p_0 \prod_{i=1}^{n} \frac{\lambda_{i-1}}{\mu_i} \tag{3.10}$$

where $p_i, (0 \leq i \leq n)$ is the probability of having i customers in the system.

We can state μ_n as follows: if there are more than c customers in the system, all the c servers are busy and each is serving at a mean rate of μ, and the mean system output is $c\mu$. When there are fewer than c customers in the system, $n < c$, only n of the c servers are busy and the system output will be $n\mu$. Therefore, we can write the statement for μ_n as

$$\mu_n = \begin{cases} n\mu; & (0 \leq n < c) \\ \\ c\mu; & (c \leq n \leq K) \end{cases} \tag{3.11}$$

Noting the fact that $\lambda_n = \lambda$ for all n as expressed in the coefficients,

$$\lambda_n = \begin{cases} \lambda; & (0 \leq n < K) \\ \\ 0; & (n \geq K) \end{cases} \tag{3.12}$$

, we can utilize equation 3.10 and equation 3.11 to determine the steady state probabilities in an M/M/c/K queue as follows:

$$p_n = \begin{cases} \frac{\lambda^n}{n!\mu^n} p_0 & (0 \leq n \ < c) \\ \\ \frac{\lambda^n}{c^{n-c}c!\mu^n} p_0 & (c \leq n \ \leq K) \end{cases} \tag{3.13}$$

In order to get p_0, we must use the boundary condition $\sum_{n=0}^{K} p_n = 1$, which gives

$$p_0 \left[\sum_{n=0}^{c-1} \frac{\lambda^n}{n!\mu^n} + \sum_{n=c}^{K} \frac{\lambda^n}{c^{n-c}c!\mu^n} \right] = 1 \tag{3.14}$$

Let us define $r = \lambda/\mu$, and $\rho = r/c = \lambda/c\mu$, then we can rewrite the above equation as

$$p_0 \left[\sum_{n=0}^{c-1} \frac{r^n}{n!} + \sum_{n=c}^{K} \frac{r^n}{c^{n-c}c!} \right] = 1 \tag{3.15}$$

Hence,

$$p_0 = \left[\sum_{n=0}^{c-1} \frac{r^n}{n!} + \sum_{n=c}^{K} \frac{r^n}{c^{n-c}c!} \right]^{-1} \tag{3.16}$$

Also, let us consider the series

$$\sum_{n=c}^{K} \frac{r^n}{c^{n-c}c!} = \frac{r^c}{c!} \sum_{n=c}^{K} (\frac{r}{c})^{n-c} \tag{3.17}$$

$$= \frac{r^c}{c!} \sum_{i=0}^{K-c} (\frac{r}{c})^i \tag{3.18}$$

$$= \begin{cases} \frac{r^c}{c!} \frac{1-\rho^{K-c+1}}{(1-\rho)}, & (r/c = \rho \neq 1) \\ \\ \frac{r^c}{c!}(K-c+1), & (\rho = 1) \end{cases} \tag{3.19}$$

Then, we can write p_0 as

$$p_0 = \begin{cases} 1/\left[\sum_{n=0}^{c-1} \frac{r^n}{n!} + \frac{r^c}{c!}\left(\frac{1-\rho^{K-c+1}}{1-\rho}\right)\right], & (r/c = \rho \neq 1) \\ \\ 1/\left[\sum_{n=0}^{c-1} \frac{r^n}{n!} + \frac{r^c}{c!}(K-c+1)\right], & (r/c = \rho = 1) \end{cases} \tag{3.20}$$

We can derive the measures of effectiveness for the M/M/c/K queue such as the expected queue size, L_q, the expected line length, L the expected waiting time in queue, W_q and the expected waiting time in line, W. In order to derive L_q, we only consider the p_n's when all the servers are busy, $n \geq c$.

By definition,

$$L_q = \sum_{n=c}^{K} (n-c)p_n \tag{3.21}$$

$$\begin{aligned} &= \frac{p_0}{c!}\sum_{n=c}^{K}\frac{(n-c)r^n}{c^{n-c}} \\ &= \frac{p_0(c\rho)^c\rho}{c!}\sum_{n=c}^{K}(n-c)\rho^{n-c-1} \\ &= \frac{p_0(c\rho)^c\rho}{c!}\sum_{m=1}^{K-c} m\rho^{m-1} \\ &= \frac{p_0(c\rho)^c\rho}{c!}\frac{d}{d\rho}\left[\frac{1-\rho^{K-c+1}}{1-\rho}\right] \\ &= \frac{p_0(c\rho)^c\rho}{c!((1-\rho)^2}\left[1-\rho^{K-c+1}-(1-\rho)(K-c+1)\rho^{K-c}\right] \end{aligned} \tag{3.22}$$

For $\rho = 1$ we apply L'Hôpital's rule twice to equation 3.22.

In order to obtain the expected line length, L, we recall equation 3.21,

$$\begin{aligned} L_q &= \sum_{n=c}^{K}(n-c)p_n \\ &= \sum_{n=c}^{K} np_n - c\sum_{n=c}^{K} p_n \\ &= \sum_{n=0}^{K} np_n - \sum_{n=0}^{c-1} np_n - c\sum_{n=c}^{K} p_n \end{aligned}$$

$$= \quad L - \sum_{n=0}^{c-1} np_n - c\left(1 - \sum_{n=0}^{c-1} p_n\right)$$
$$= \quad L - \sum_{n=0}^{c-1} (n-c)p_n - c$$

Therefore,

$$L = L_q + c - \sum_{n=0}^{c-1} (c-n)p_n \tag{3.23}$$

or

$$L = L_q + c - p_0 \sum_{n=0}^{c-1} \frac{(c-n)(\rho c)^n}{n!} \tag{3.24}$$

We can use Little's formula to obtain the expected values for waiting times, W and W_q as follows:

$W = L/\lambda', \lambda' = \lambda(1 - p_K)$

and $W_q = W - 1/\mu$

or W_q =L_q/μ'

In our simple example $\lambda = 4$, $\mu = 3$, $c = 3$ and $K = 5$. Working to 4 decimal places, $r = \lambda/\mu = 4/3 = 1.3333$, $\rho = r/c = 1.3333/3 = 0.4444$.

We use equation 3.20 to find p_0, thus,

$p_0 = 1/\left[\sum_{n=0}^{c-1} \frac{r^n}{n!} + \frac{r^c}{c!}\left(\frac{1-\rho^{K-c+1}}{1-\rho}\right)\right]$, since $r/c = \rho \neq 1$)

$= 1/\left[\sum_{n=0}^{3-1} \frac{1.3333^n}{n!} + \frac{1.3333^3}{3!}\left(\frac{1-0.4444^{5-3+1}}{1-0.4444}\right)\right]$

$= 1/\left[\sum_{n=0}^{2} \frac{1.3333^n}{n!} + \frac{2.3702}{6}\left(\frac{1-0.0878}{0.5556}\right)\right]$

$= 1/\left[1 + 1.3333 + 0.8888 + 0.6486\right]$

$= 1/3.8707$

$p_0 = 0.2584$

We can now use equation 3.13 to find the remaining $p_n's$, thus

$p_1 = (1.3333 * 0.2584)/1$

$p_1 = 0.3445$

$p_2 = (1.3333^2 * 0.2584)/2!$

$p_2 = 0.2297$

$p_3 = (1.3333^3 * 0.2584)/3!$

$p_3 = 0.1021$

$p_4 = (1.3333^4 * 0.2584)/(3^1 * 3!)$

$= 0.8166/18$

$p_4 = 0.0454$

$p_5 = (1.3333^5 * 0.2584)/(3^2 * 3!)$

$= 1.0888/54$

$p_5 = 0.0202$

These results correspond with the ones we got for $Prob(X1 = 0), Prob(X1 = 1), Prob(X1 = 2), Prob(X1 = 3), Prob(X1 = 4)$ and $Prob(X1 = 5)$ in Möbius and Eqsp.

Now using equation 3.21 we can find the expected queue length, L_q.

$L_q = \sum_{n=c}^{K} (n - c)p_n$

$= \sum_{n=3}^{5} (n - 3)p_n$

$= (0 * 0.1021) + (1 * 0.0454) + (2 * 0.0202)$

$L_q = 0.0858$

In order to obtain the expected line length, L, we recall equation 3.24,

$L = L_q + c - p_0 \sum_{n=0}^{c-1} \frac{(c-n)(\rho c)^n}{n!}$

$= 0.0858 + 3 - 0.2584(\sum_{n=0}^{3-1} \frac{(3-n)(3*0.4444)^n}{n!})$

$= 3.0858 - 0.2584(\sum_{n=0}^{2} \frac{(3-n)(1.3332)^n}{n!})$

$= 3.0858 - 0.2584(3 + 2.6664 + 0.8887)$

$= 3.0858 - 0.2584(6.5551)$

$L = 1.3920$

Again, this corresponds with the value of $Exp(X1)$ obtained in Möbius and Eqsp.

Chapter 4

Experiment Setup and Methodology

The experiments are planned to establish the optimal staffing levels and the recommended mix of agents in systems with specialized and flexible agents. The experiments are designed to address the following research questions:

Is there any benefit to system performance of introducing cross-trained agents? If yes, to what extent?

In order to answer this question, we first consider a system with only specialized agents and measure its performance and cost in terms of the cost of the hired agents at work, lost customers and customers waiting in queue. We then remove one specialized agent at a time and introduce a flexible agent and compare the resulting system performance and cost. If the system is more efficient in overall performance and cost with the introduction of flexible agents then there is a benefit. In the two-category customer model there are no partially cross-trained servers. A server is either specialized or fully flexible. In the three-category customer model, however, there are partially cross-trained servers. These are agents who can speak two languages.

Since the cost of hiring a flexible agent is higher than that of hiring a specialized agent and given that there is a limit to the total number of agents we can have at work at any given time, we envisage that there is likely a point at which the introduction of any more flexible agents deteriorates the efficiency of the system. We watch out for the point at which this situation occurs. The combination of servers that yields a comparatively low cost becomes the desired staffing level. We investigate this for both the two-category customer model and three-category customer model.

How is the staffing level in such queueing systems affected when there is an unevenness in the traffic load of the system? What is the effect on cost? In order to

answer this question, we use distinct arrival rates for each of the customer categories. In other words, we conduct experiments where $\lambda_A \neq \lambda_B \neq \lambda_C$ and investigate if there is a significant effect on the system cost and agent-staffing patterns. Specifically, we will consider the case where $\lambda_A > \lambda_B > \lambda_C$ or $\lambda_A < \lambda_B < \lambda_C$.

The experiments for determining system performance measures of probability of lost customers, mean number of customers in queue and waiting time of customers is conducted in Möbius. Figure 4.1 shows the framework for the Stochastic Activity Network of a three-customer category model. Möbius provides an easy way of reducing the framework to a two-customer category model when one assigns a parametric value of zero to the arrival rate of one of the customer streams, $\lambda_C = 0$. Similarly, by assigning a value of zero to the arrival rates of two of the customer streams such that $\lambda_B = 0$ and $\lambda_C = 0$, the framework is reduced to a one-customer category model which, obviously, is the single $M/M/c/K$ queue model. Furthermore, it is possible to expand the framework to study systems where there are more than three categories of customers. In this case, one needs to appropriately introduce more gates, places and activities properly interconnected by arcs. The agents— specialized, partially cross-trained and fully cross-trained are represented with circles which are called "places" in Möbius. The thin vertical lines show instantaneous activities which means that there is no time delay when, for example, a customer moves from the queue to a server. Refer to pages 25 - 28 for the explanation of the graphical symbols.

We reasonably assign labour cost to each category of agents. We also allocate cost to each lost customer and each unit time waited in the queue by a customer. We then calculate the total system cost, T_C. Using the optimization model, we minimize this cost.

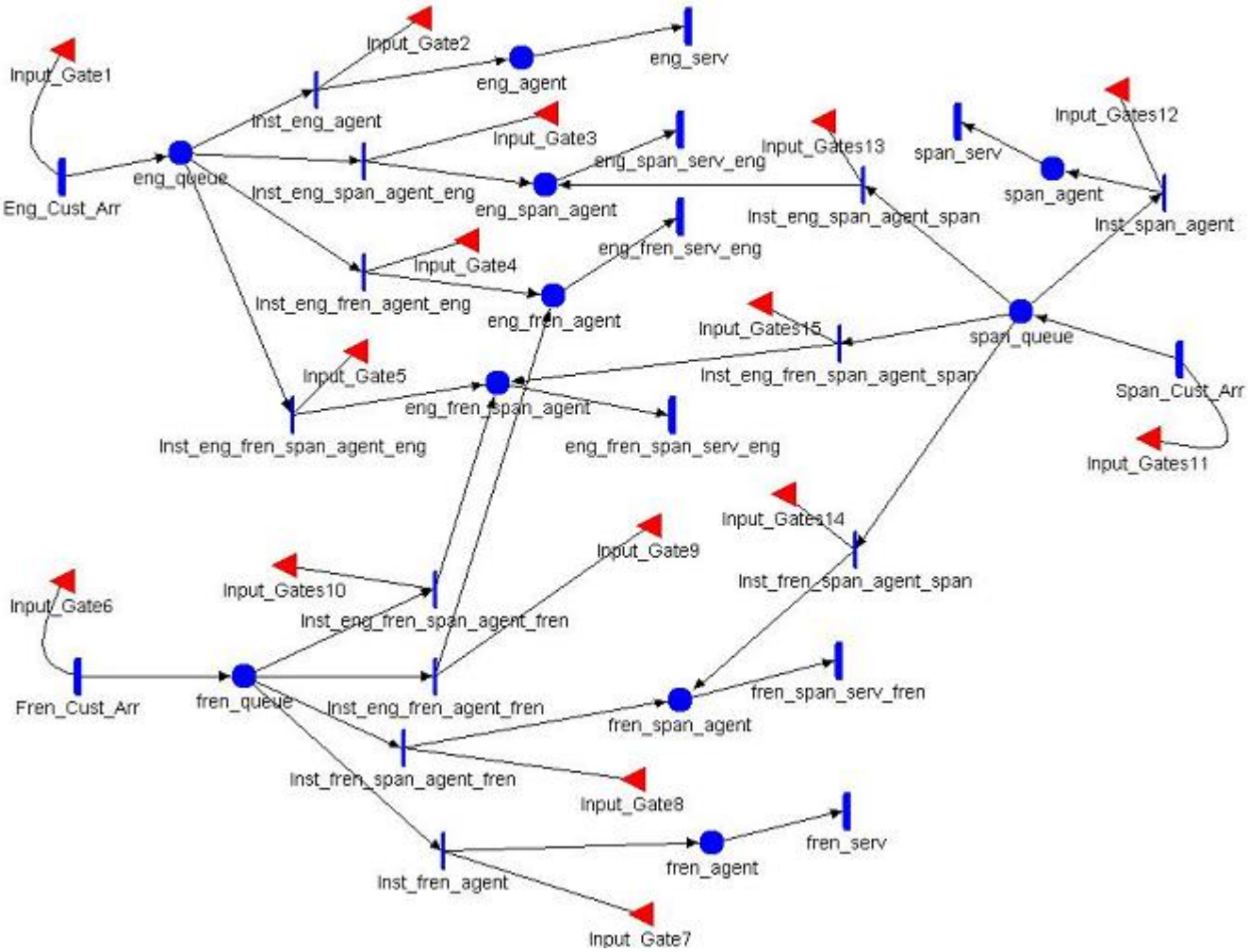


Figure 4.1: The Queueing Model Framework

4.1 Two-Customer Category Cost Ingredients and Experiments

Obviously, there are various factors that can affect cost in a queueing system. The cost arising from the loss of customers, S_L, is the major cost factor in the system, because in most establishments it is the customers that buy goods and services and generate revenue for the firm. The revenue obtained, among other things, is used to remunerate the agents that work in the establishment. In order for the company to make profit, revenue should be more than the expenses incured. When a customer is lost, the revenue that should have been generated by this customer is lost and this is the cost of the lost customer. In this study, we let the cost of loss be 22.00 dollars per customer lost.

Another cost ingredient that can be considered is S_W, the wait cost or the cost due to customers waiting in the queue. The wait cost can impact on the reputation of the organization if the customers have to wait too long before receiving service. In competitive environments, the establishment can lose clients to other competitors due to customer dissatisfaction arising from long waits. Management usually will set a wait time threshold to ensure that customers do not wait too long before receiving service. In any case, the cost assigned to a customer waiting in the queue per unit time should only be a small fraction of the cost of losing a customer per unit time. In some circumstances where the service obtained is worth the wait, it might not be necessary to assign a waiting cost to the system. Thus, we can assume a zero wait time cost in such situations. Generally, the cost assigned to a customer lost per unit time is more than the cost associated with a customer waiting in the queue per unit time. For the purpose of this study, we assign a waiting cost of 3.00 dollars for a unit time waited by a customer in queue.

The cost of agents is another important cost ingredient in queueing systems. It is one of the factors that can help us to determine if cross-trained agents can be used in such systems. The availability of cross-trained agents, their wages or cost

of hiring and their service rate when compared with their specialized counterparts are important factors to consider when making agent-employment decisions. If the cost of hiring flexible servers is excessively high and their service rate is excessively low, using specialized servers will give lower cost and better system performance. Conversely, if the cost of hiring a cross-trained agent and a specialized agent is about the same or their service rates are almost equal, then there may be benefits of employing cross-trained agents. In practice, it is most likely that the cost of employing a cross-trained agent is higher than the cost of employing a specialized agent, $S_1 \leq S_2 \leq S_3$, because it is easier to find agents that can speak only one of English, French and Spanish than those that can speak all three. Similarly, the service rate of a specialized agent is most likely higher than that of a cross-trained agent, $\mu_3 \leq \mu_2 \leq \mu_1$, because due to constant practice and specialization, a dedicated agent is likely to be faster than a flexible agent. We note that the wages of the agents per time unit S_1, S_2 and S_3, should also be less than the cost of losing a customer per time unit in an effort to ensure that the system is not running on a huge loss. However, one can keep $S_1 = S_2 = S_3$ and consider $\mu_3 < \mu_2 < \mu_1$ and check the benefits of cross-training. This is possible because management can bargain to place all agents on equal wage.

It is also possible to keep $\mu_3 = \mu_2 = \mu_1$ and assume $S_1 < S_2 < S_3$. In this case, the service rate of all agents are equal and cross-trained agents are more expensive than specialized agents. Due to the difference between the cost of a cross-trained agent and a specialized agents our guess is that there may be some points where hiring cross-trained agents are not beneficial. We will investigate the benefits of cross-training in this scenario.

If $S_1 = S_2 = S_3$ and $\mu_3 = \mu_2 = \mu_1$, it is simple to show that the best option will be to employ only cross-trained agents. Their flexibility and comparative low cost will impact positively on system cost and performance. Since this scenario is trivial, we will not investigate it.

In both the two-category customer model and the three-category customer model our first set of experiments and analysis will be done keeping agent costs constant.

In the second set of experiments and analysis we will keep service rates constant.

4.1.1 Two-Customer Category with Equal Arrival Rates

In a two-customer category with equal arrival rates, we have two categories of customers, A and B whose arrival rates λ_A and λ_B are equal. An agent in this type of system is either specialized or fully cross-trained. Given our assumptions that the service rates of all specialized agents are equal and the service rates of all cross-trained agents are equal in the case of two streams of customers, we expect the reward measures of each of the categories to give the same result when the agents are evenly allocated to serve the two categories of customers. Where an even number of specialized agents are used, the best allocation at such points involves balancing them equally between the two categories.

In finding the optimal agent allocation, two decision situations come in mind. The first situation arises when the person in charge of agent allocation is constrained by the total number of agents available. Given that the agents available could be specialized or cross-trained he has to find out how to combine these agents to get the optimal cost. This is similar to finding the local optimum at the given constraint. The second situation is when there is no constraint in the number of agents and there is the freedom to choose. Here, he seeks to find the agent mix that will result in the global optimal cost of the system.

Two-Customer Category: Equal Arrival Rates, Equal Agent Cost and Varying Service Rates

In this study, the assumptions $\lambda_A = \lambda_B$, $S_1 = S_2$ and $\mu_2 < \mu_1$ hold. We set the arrival rates $\lambda_A = \lambda_B = 2.0$, and the costs $S_1 = S_2 = 10$ dollars. In varying the service rates of the cross-trained agents, we consider $\mu_2 = 0.4\mu_1$; $\mu_2 = 0.5\mu_1$; $\mu_2 = 0.75\mu_1$ and $\mu_2 = 0.9\mu_1$ where $\mu_1 = 2.0$. Given these conditions, we guess that more cross-trained agents will be involved in the best agent mix when $\mu_2 = 0.9\mu_1$ than when $\mu_2 = 0.4\mu_1$. We investigate how much gain is achieved in cross-training at the other values of μ_2 and find the optimal agent combination.

Two-Customer Category: Equal Arrival Rates, Equal Service Rates and Varying Agent Cost

In this case, $\lambda_A = \lambda_B$, $\mu_1 = \mu_2$ and $S_1 < S_2$. We set the arrival rates $\lambda_A = \lambda_B = 2.0$ and the service rates $\mu_1 = \mu_2 = 2.0$. We study the behaviour of the system when the cost of a cross-trained agent is higher than the cost of a specialized agent as follows: $S_2 = 1.2S_1$; $S_2 = 1.5S_1$; $S_2 = 1.8S_1$ and $S_2 = 2S_1$ where $S_1 = 10$ dollars. Our guess here is that more cross-trained agents will be used for optimal allocation when S_2 approaches S_1, $(S_2 = 1.2S_1)$ than when it is much greater than S_1, $(S_2 = 2S_1)$.

4.1.2 Two-Customer Category with Different Arrival Rates

The same sets of experiments above are repeated except that we keep $\lambda_A > \lambda_B$. We guess that for optimality, the effective number of agents serving category A customers should be more than those serving category B customers.

Two-Customer Category: Different Arrival Rates, Varying Service Rates and Equal Agent Cost

In this case, $\lambda_A > \lambda_B$, $\mu_2 < \mu_1$ and $S_1 = S_2$. We set $\lambda_B = 2.0$, $\lambda_A = 2\lambda_B$, and $S_1 = S_2 = 10$ dollars. We study the system for $\mu_2 = 0.4\mu_1$; $\mu_2 = 0.5\mu_1$; $\mu_2 = 0.75\mu_1$ and $\mu_2 = 0.9\mu_1$ with $\mu_1 = 2.0$. At any optimal solution, we suspect the possibility of using more cross-trained agents for $\mu_2 = 0.9\mu_1$ than for $\mu_2 = 0.4\mu_1$. We investigate how the cross-trained agents and specialized agents are mixed for optimal solutions at the other intermediate values of μ_2.

Two-Customer Category: Different Arrival Rates, Varying Agent Cost and Equal Service Rates

Here, $\lambda_A > \lambda_B$, $S_1 < S_2$ and $\mu_1 = \mu_2$. We set $\lambda_B = 2.0$, $\lambda_A = 2\lambda_B$, and $\mu_1 = \mu_2 = 2.0$. We study the system for the cost of agents: $S_2 = 1.2S_1$, $S_2 = 1.5S_1$, $S_2 = 1.8S_1$, $S_2 = 2S_1$ where $S_1 = 10$ dollars. Since their service rates are equal, we guess that more cross-trained agents than specialized agents will be used to get optimal solution

when $S_2 = 1.2S_1$ than when $S_2 = 2S_1$. We evaluate the system for the other values of S_2.

4.2 Three-Customer Category Experiments

In the three-customer category, we introduce an extra class of customers, C, to the two-stream model. We now have specialized agents, partially cross-trained agents and fully cross-trained agents. The specialized agents are A-agents, B-agents, C-agents, the partially cross-trained agents are A-B agents, A-C agents and B-C agents while the fully cross-trained agents are A-B-C-agents. Thus, we have seven groups of agents. The question here is whether it is really necessary to utilize the fully cross-trained, A-B-C-agents or if the partially cross-trained agents are enough to yield optimal cost and system performance given the agents' costs and service rates.

In the three streams of customers, when N agents are employed in the system, the number of experiments conducted to arrive at the optimal solution increases exponentially when compared with the case of two stream customers. Note that $N = \sum_{i=1}^{3} n_i$ is the total number of agents used, with n_1 being the number of specialized agents, n_2 the number of partially cross-trained agents and n_3 the number of fully cross-trained agents. The increase in the number of experiments is especially true when $\lambda_A > \lambda_B > \lambda_C$ and $\mu_3 < \mu_2 < \mu_1$. Therefore, we devise a method to reduce the number of experiments significantly. Starting at $N = 1$, we perform seven experiments to get the optimum at this point. The seven experiments arise from $\sum_{i=1}^{n} \binom{n}{i}$ or $2^n - 1$ pools of agents possible in $n = 3$ streams of customers. Then, for $N = 2$, we make the optimal solution of $N = 1$ our base point and conduct seven more experiments to get the optimal solution at $N = 2$. These solutions mean the best way to allocate the agents when we are constrained by the number of agents $N = 1$, $N = 2$, and so on. We proceed in this manner until the best optimal solution is obtained at say, $N = N_{opt}$ for the given parameters. The solution at $N = N_{opt}$ gives the best way to allocate the agents when there is no limit on the availability of agents to yield the minimum cost. Thus, given number of agent in the system,

$N \geq 2$, we only need to know the optimal agent mix at $N-1$. Then we perform seven experiments to get the optimal agent mix at N. We guess that as the capacity of the system increases (as N increases), the number of specialized agents used decreases. This guess is especially trivial if there is no difference in cost of employing a cross-trained agent and a specialized agent or if there is no difference in their service rates. We point out that it is the optimal solutions at $N < N_{opt}$ that is recommended when the scheduling staff is constrained by the number of agents available for use. When he is free to use any number of agents the optimal solution at $N = N_{opt}$ is most advisable.

4.2.1 Three-Customer Category with Equal Arrival Rates

In a three-customer category with equal arrival rates, we have three streams of customers, A, B and C whose arrival rates λ_1, λ_2 and λ_3 are equal. An agent in this type of system could be specialized, partially cross-trained or fully cross-trained. Like in the two streams, we expect the reward measures of each of the categories to give the same result when the agents are evenly allocated to serve the three categories of customers. Where the number of specialized agents used is three or a multiple of three, the best allocation at such a point involves balancing the specialized agents equally among the three categories. The same balancing is also applicable when the number of partially cross-trained agents is three or a multiple of three.

Three-Customer Category: Equal Arrival Rates, Equal Agent Cost and Varying Service Rates

In this study, the assumptions $\lambda_A = \lambda_B = \lambda_C$, $S_1 = S_2 = S_3$ and $\mu_3 < \mu_2 < \mu_1$ hold. We set the arrival rates $\lambda_A = \lambda_B = \lambda_C = 2.0$, and the costs $S_1 = S_2 = S_3 = 10$ dollars. In varying the service rates of the cross-trained agents, we consider $\mu_2 = 0.4\mu_1$, $\mu_3 = 0.9\mu_2$; $\mu_2 = 0.5\mu_1$, $\mu_3 = 0.9\mu_2$; $\mu_2 = 0.75\mu_1$, $\mu_3 = 0.9\mu_2$, and $\mu_2 = 0.9\mu_1$,$\mu_3 = 0.9\mu_2$ where $\mu_1 = 2.0$. We choose $\mu_3 = 0.9\mu_2$ because since μ_2 must be greater than μ_3, we guess that if there is no gain in the use of fully cross-trained agents at this value of μ_3 then there is no advantage of using a cross-trained agent

at lower values of μ_3.

Three-Customer Category: Equal Arrival Rates, Equal Service Rates and Varying Agent Cost

In this case, $\lambda_A = \lambda_B = \lambda_C$, $\mu_1 = \mu_2 = \mu_3$ and $S_1 < S_2 < S_3$. We set the arrival rates $\lambda_A = \lambda_B = \lambda_C = 2.0$ and the service rates $\mu_1 = \mu_2 = \mu_3 = 2.0$. We study the behaviour of the system when the cost of a cross-trained agent is higher than the cost of a partially cross-trained agent and the cost of a partially cross-trained agent is higher than that of a specialized agent as follows: $S_2 = 1.2S_1, S_3 = 1.25S_2$; $S_2 = 1.5S_1, S_3 = 1.25S_2$; and $S_2 = 2S_1, S_3 = 1.25S_2$ where $S_1 = 10$ dollars.

4.2.2 Three-Customer Category with Different Arrival Rates

The same sets of experiments above are repeated except that we keep $\lambda_A > \lambda_B > \lambda_C$. We guess that for optimality, the effective number of agents serving category A customers should be more than that serving category B customers and the number of agents attending to category B customers should be more than that serving category C customers.

Three-Customer Category: Different Arrival Rates, Varying Service Rates and Equal Agent Cost

In this case, $\lambda_A > \lambda_B > \lambda_C$, $\mu_3 < \mu_2 < \mu_1$ and $S_1 = S_2 = S_3$. We set $\lambda_C = 2.0$, $\lambda_B = 2\lambda_C$, $\lambda_A = 3\lambda_C$, and $S_1 = S_2 = S_3 = 10$ dollars. We study the system for $\mu_2 = 0.4\mu_1$, $\mu_3 = 0.9\mu_2$; $\mu_2 = 0.5\mu_1$, $\mu_3 = 0.9\mu_2$; $\mu_2 = 0.75\mu_1$, $\mu_3 = 0.9\mu_2$, and $\mu_2 = 0.9\mu_1$,$\mu_3 = 0.9\mu_2$ where $\mu_1 = 2.0$.

Three-Customer Category: Different Arrival Rates, Varying Agent Cost and Equal Service Rates

Here, $\lambda_A > \lambda_B > \lambda_C$, $S_1 < S_2 < S_3$ and $\mu_1 = \mu_2 = \mu_3$. We set $\lambda_C = 2.0$, $\lambda_B = 2\lambda_C$, $\lambda_A = 3\lambda_C$, and $\mu_1 = \mu_2 = \mu_3 = 2.0$. We study the system for the cost of agents:

$S_2 = 1.2S_1, S_3 = 1.25S_2$; $S_2 = 1.5S_1, S_3 = 1.25S_2$; and $S_2 = 2S_1, S_3 = 1.25S_2$ where $S_1 = 10$ dollars.

Chapter 5

Results

We now evaluate the results obtained from each of the experiment segments. These are the experiments using various service rate factors and cross-trained agent cost factors.

5.1 Two-Customer Category: Equal Arrival Rates, Equal Agent Cost and Varying Service Rates

We define the cross-trained agent service rate factor as the ratio of the service rate of a cross-trained agent to the service rate of a specialized agent, or μ_2/μ_1. This ratio is less than or equal to one. If the ratio is one, it means that a cross-trained agent is as fast as a specialized agent. A lower-than-one ratio means that the cross-trained agent is slower than a specialized agent. We find that for all values of this ratio, the system cost reduces as the system capacity increases until after the optimum point, $N = N_{opt}$, when additional increase in the number of agents increases system cost. The system cost is the optimal at the point $N = N_{opt}$. The best cost at the various points of $N, 1 \leq N < N_{opt}$ only reflects the optimum when we are constrained by the number of agents allowed in the system. Even if we are not constrained by the number of agents available for use, the best cost at $N > N_{opt}$ will start to deteriorate and can not be better than the optimal at $N = N_{opt}$. The optimal cost for lower service rate factors is higher than the optimal cost for higher service rate factors. As shown in Figure 5.1 and Figure 5.3 when the service rate factor is 0.4 the optimal system cost is 44.87 dollars but when a higher service rate factor of 0.9 is used this

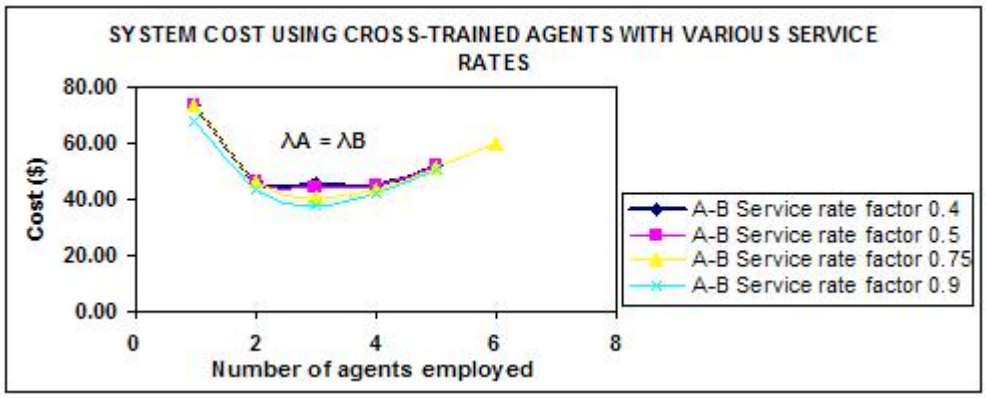


Figure 5.1: Agent Utilization and System Cost: $\lambda_A = \lambda_B$, $S_1 = S_2$

cost reduces to 37.55 dollars. The implication of this is that cross-trained agents with high service rates are more flexible, more preferable and impact more positively on system performance than ones with low service rates. The optimum number of agents used can be read from figure 5.1. For example, for the service rate factor of 0.9, three agents are used to achieve optimal solution. For the service rate factor of 0.4, four agents are used to achieve optimal system cost. However, the service rate factor affects the way specialized agents and cross-trained agents are mixed to yield the optimal system performance. Let us consider Figure 5.2. When the service rate factor is 0.4, the use of only specialized agents (four of them) gives the best system performance and no cross-trained agent is required. When the service rate factor is 0.9, all the three agents used to achieve the optimal system cost are cross-trained. This means that when the service rate factor is low fewer cross-trained agents are involved in the agent combination that yields the best system performance. As the service rate factor increases the number of cross-trained agents required for optimal cost increases. The figure also shows that for a given service rate factor, equal percentage of specialized A-agents and specialized B-agents are used when the optimal system performance is achieved. This confirms our conjecture that there is the need to balance the specialized agents for each of the customer categories when the customer arrival rates are equal in the two categories.

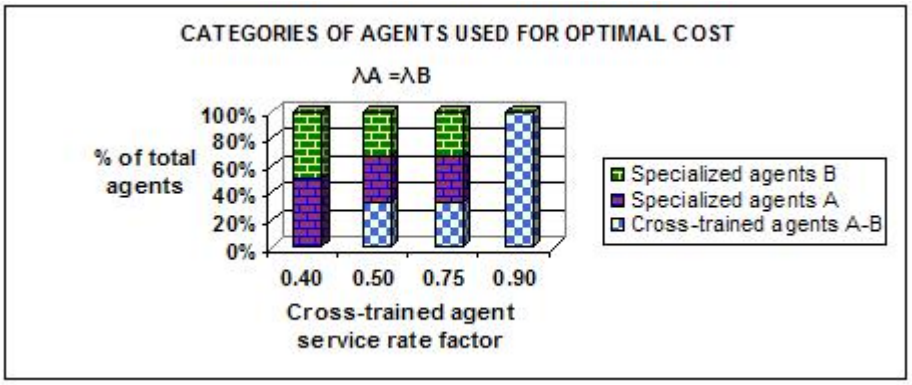


Figure 5.2: Optimal Agent Mix: $\lambda_A = \lambda_B$, $S_1 = S_2$

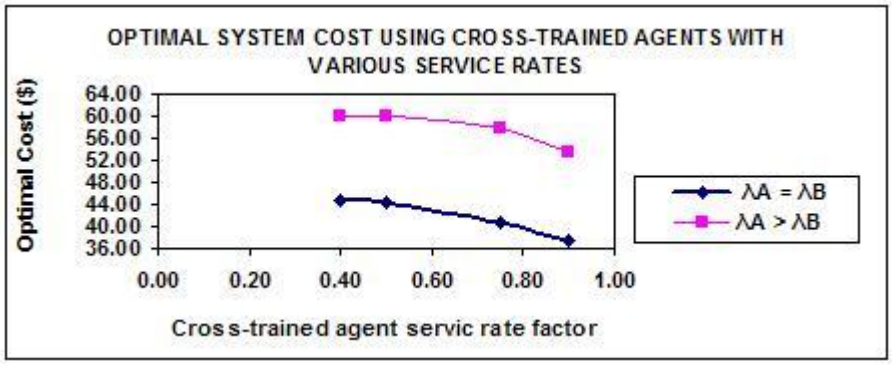


Figure 5.3: Optimal System Cost: $S_1 = S_2$

5.2 Two-Customer Category: Different Arrival Rates, Varying Service Rates and Equal Agent Cost

The system cost pattern here is similar to the pattern when the arrival rates of the two categories of customers are equal. The higher the number of agents used in the system, the lower the system cost until the optimum point beyond which any additional agent used results in a higher system cost. Similarly, the optimal cost for lower service rate factors is higher than the optimal cost for higher service rate factors. For example, as shown in figures 5.4 and 5.3, when the service rate factor is 0.4 the optimal system cost is 59.90 dollars but when a higher service rate factor of 0.9 is used this cost reduces to about 53.51 dollars. The service rate factor also affects the way specialized agents and cross-trained agents are mixed to yield the optimal

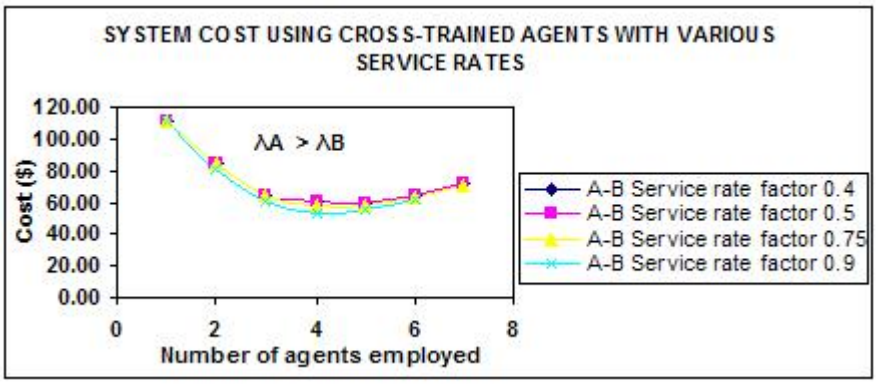


Figure 5.4: Agent Utilization and System Cost: $\lambda_A > \lambda_B$, $S_1 = S_2$

system performance. When the service rate factor is 0.4, the use of only specialized agents gives the best system performance and no cross-trained agent is required as shown in figure 5.5. When the service rate factor is 0.9, more cross-trained agents than specialized agents are used for optimal cost. As the service rate factor increases, the number of cross-trained agents required for optimal cost increases. Among all the service rate factors considered, there is no case when only cross-trained agents are used for optimal system cost. Instead, there must be the use of specialized agents especially the ones that pertain to the category that has the higher arrival rate in order to get a combination that yields the best system performance. Unlike the equal arrival rate case, more specialized A-agents are used than specialized B-agents. This is due to the fact that A customers arrival rate is higher than B customers arrival rate. Therefore, more specialized A-agents than specialized B agents are required to take care of the higher A customer arrivals.

5.3 Two-Customer Category: Equal Arrival Rates, Equal Service Rates and Varying Agent Cost

We define the cross-trained agent cost factor as the ratio of the cost of a cross-trained agent A-B per time unit to the cost of a specialized agent-A per time unit, or S_2/S_1. This ratio is greater or equal to one. If it is one it means that $S_1 = S_2$. In other words, the cost of the cross-trained agent per time unit is equal to the cost of a specialized

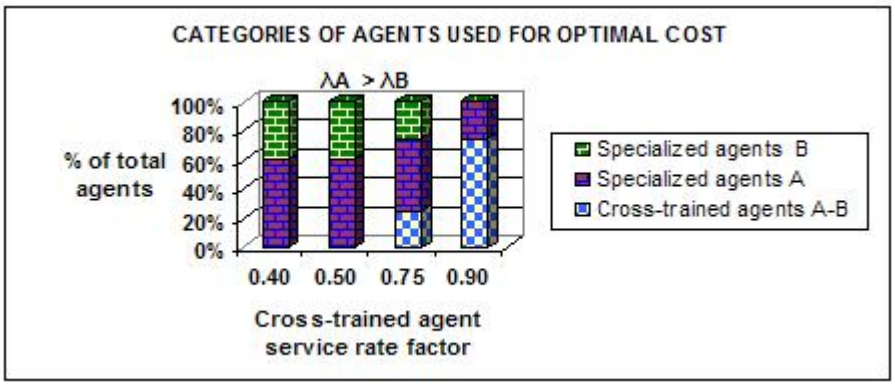


Figure 5.5: Optimal Agent Mix: $\lambda_A > \lambda_B$, $S_1 = S_2$

agent per time unit. A ratio greater than one means that the cross-trained agent is more expensive than a specialized agent.

Irrespective of the cross-trained agent cost ratio that we are looking at, the system cost decreases as the total number of agents employed increases until the optimum cost is reached. After this point, the introduction of more agents into the system results in an increase in the system cost. This is shown in Figures 5.6 and 5.8. We also find that as the ratio increases, the percentage of cross-trained agents employed for optimal system performance decreases as shown in Figure 5.7. When the ratio is one, only cross-trained agents are used to obtain the minimum cost. It is at this ratio that the flexibility of cross-trained agents fully comes into play and is clearly appreciated. At the ratio of 2.0, no cross-trained agents are used. When the ratio is 1.2, a third of the agents employed for optimal system cost is cross-trained and the remaining two-thirds are specialized and equally distributed as specialized A-agents and specialized B-agents.

5.4 Two-Customer Category: Different Arrival Rates, Varying Agent Cost and Equal Service Rates

When the arrival rates of the two streams of customers are different, the system cost also reduces as more agents are employed until the optimal point after which the use of any more agents results in an increase in cost. The optimal cost is different

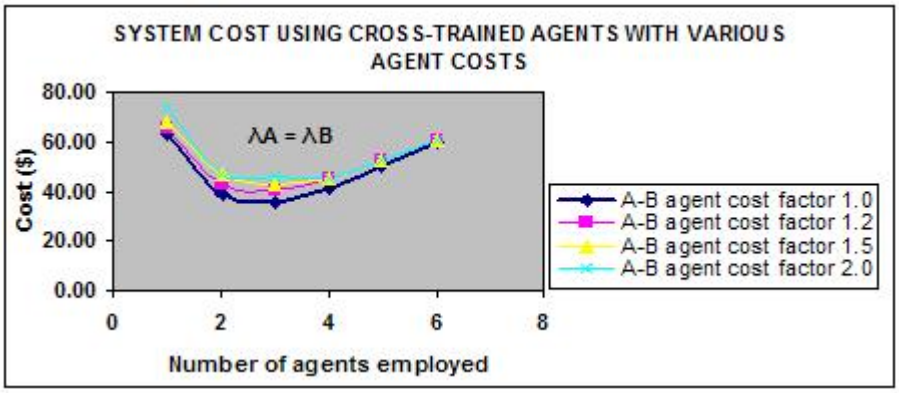


Figure 5.6: Agent Utilization and System Cost: $\lambda_A = \lambda_B$, $\mu_1 = \mu_2$

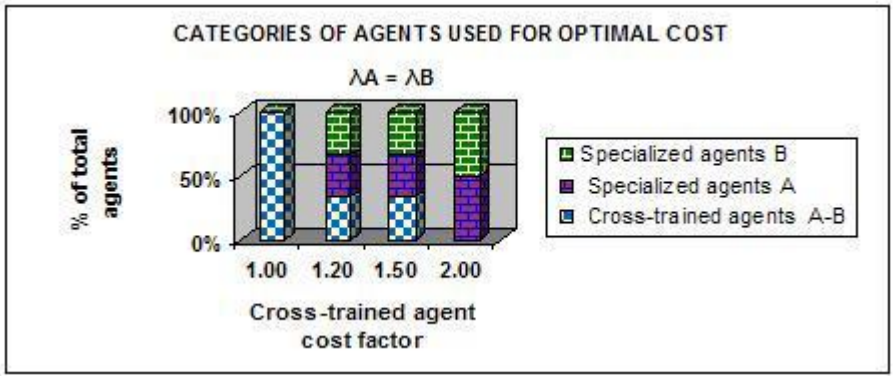


Figure 5.7: Optimal Agent Mix: $\lambda_A = \lambda_B$, $\mu_1 = \mu_2$

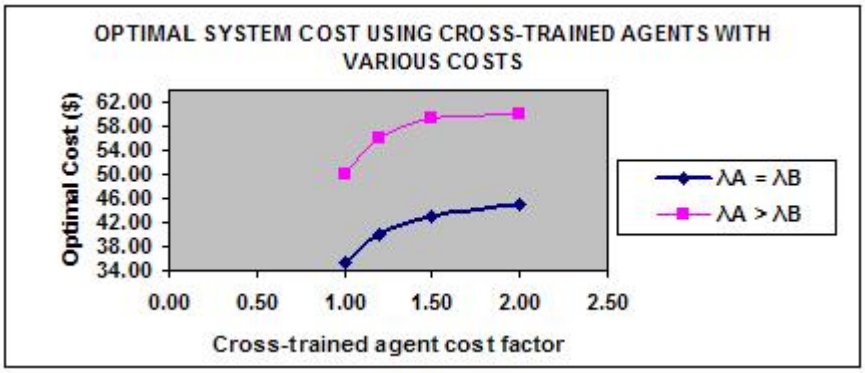


Figure 5.8: Optimal System Cost: $\mu_1 = \mu_2$

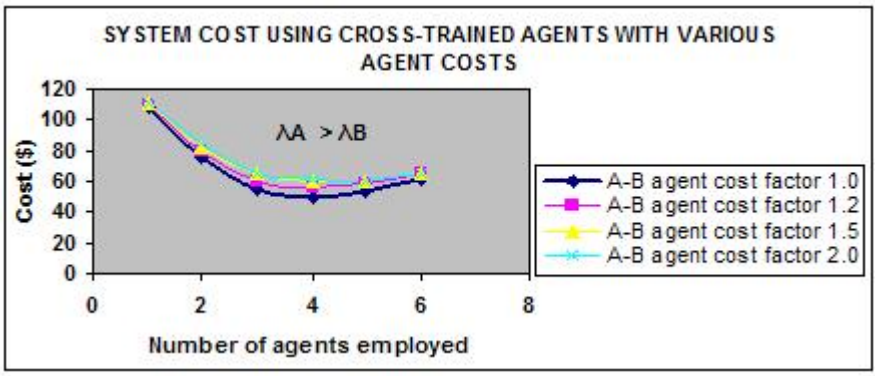


Figure 5.9: Agent Utilization and System Cost: $\lambda_A > \lambda_B$, $\mu_1 = \mu_2$

for each cross-trained agent cost factor considered. In fact, the optimal cost of the system increases as the cross-trained agent cost factor increases. For example when the cross-trained agent cost factor is one, the optimal cost is 49.97 dollars and when the cross-trained agent cost factor is 1.5, the optimal cost increases to 59.41 dollars as can be seen in figures 5.9 and 5.8.

In terms of the agent distribution at optimal system cost, we find that the percentage of agents used that are cross-trained decreases with increase in the cross-trained agent cost factor. A hundred percent of the agents used are cross-trained for a cross-trained agent cost factor of 1.0 while no cross-trained agent is used at all for cross-trained agent cost factor of 2.0. When specialized agents are used, the allocation is biased to specialized A-agents because of the higher arrival rate of A-customers compared with that of B-customers. We can see this in figure 5.10. Consider the case when the cross-trained agent cost factor is 1.5. Twenty-five percent of the agents is cross-trained and seventy-five percent is specialized with fifty percent being specialized A-agents.

5.5 Three-Customer Category: Equal Arrival Rates, Equal Agent Cost and Varying Service Rates

The partially cross-trained agent service rate factor for the three-customer category is the ratio of the service rate of a partially cross-trained agent to the service rate

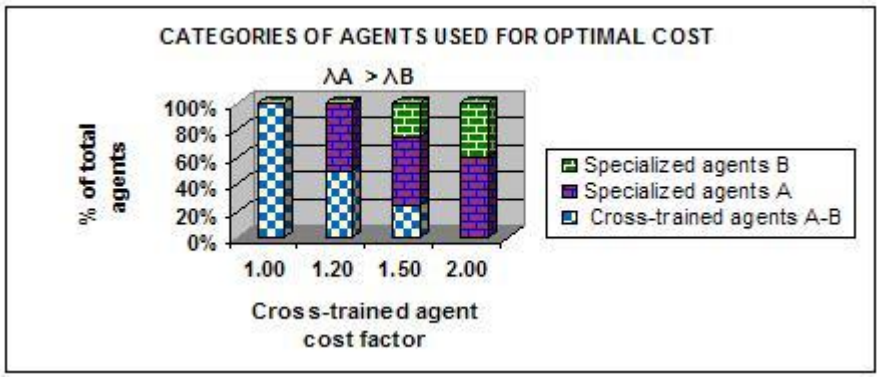


Figure 5.10: Optimal Agent Mix: $\lambda_A > \lambda_B$, $\mu_1 = \mu_2$

of a specialized agent, or μ_2/μ_1. It has the same value as the cross-trained agent service rate factor for the two-customer category. The difference in terminology is necessary to differentiate between the fully cross-trained agents and partially cross-trained agents that come into play in the three-customer category. Similar to the two customer category case, the partially cross-trained agent service rate factor is less than or equal to one. If $\mu_2/\mu_1 = 1$, it means that a partially cross-trained agent is as fast as a specialized agent. A lower-than-one ratio means that the partially cross-trained agent is slower than a specialized agent. We keep the service rate of a fully cross-trained agent, μ_3, as a fraction of the service rate of a partially cross-trained agent, μ_2, such that $\mu_3 = 0.9\mu_2$. We take 0.9 because it is a service rate factor that is reasonable and high enough for a fully cross-trained agent to make it comparable and competitive with its partially cross-trained counterpart. This means that if the use of a fully cross-trained agent with this service rate does not yield an optimal system cost, then it is not efficient and should be avoided in the three customer stream.

We find that just like in the two-customer stream case, no matter the value of the partially cross-trained agent service rate factor, the system cost reduces as the system capacity increases until the optimum point when additional increase in the number of agents increases system cost. The optimal point is different for various service rate factors. The optimal system cost for lower service rate factors is higher than the optimal cost for higher service rate factors as shown in Figures 5.11 and 5.13. For example, when the service rate factor is 0.4 the optimal system cost is

67.30 dollars, but when a higher service rate factor of 0.75 is used this cost reduces to 60.40 dollars. The implication of this is that cross-trained agents with high service rate are preferable and impact more positively on system performance.

The service rate factor affects the way specialized agents, partially cross-trained agents and cross-trained agents are mixed to yield optimal system performance. Consider figure 5.12. Similar to the two-customer category model, when the service rate factor is 0.4, the use of only specialized agents gives the best system performance and no partially or fully cross-trained agent is required. The specialized agents are distributed equally among type-A customers, type-B customers and type-C customers. This is so because the arrival rates of all the three streams of customers are equal. At service rate factors of 0.5 and 0.75, fully cross-trained agents are used in combination with specialized agents at a ratio of 2:3 to give the best system cost. Again the specialized agents are balanced among the three customer types. That there is no partially cross-trained agent involved in the optimal solution is not a surprise because of the service rate of the fully cross-trained agents at $\mu_3 = 0.9\mu_2$ which enhances their flexibility and competitiveness compared with the partially cross-trained agents. At a service rate factors of 0.5, $\mu_2 = 1.0$ and $\mu_3 = 0.9$. With a service rate factor of 0.75, $\mu_2 = 1.50$ and $\mu_3 = 1.35$. If the fully cross-trained agents had a lower service rate, their speed would reduce and this will impact negatively on their flexibility, and probably make the partially cross-trained agents preferable.

Similar to the two customer category case where the flexibility of the cross-trained agents significantly comes into play when the service rate factor is 0.9, we see that no specialized agent is used for optimal cost for this service rate factor in the three stream model. Instead, only partially cross-trained and fully cross-trained agents are utilized, with partially cross-trained agents making up seventy-five percent of the total agents and the fully cross-trained being twenty-five percent. Again, the partially cross-trained are equally allocated as A-B-agents, A-C-agents and B-C-agents. The reason is that the arrival rates, λ_A, λ_B and λ_C are equal.

When the service rate factor is 0.9, only cross-trained agents are used for optimal cost. This means that when the service rate factor is low, fewer cross-trained agents

Figure 5.11: Agent Utilization and System Cost: $\lambda_A = \lambda_B = \lambda_C$, $S_1 = S_2 = S_3$

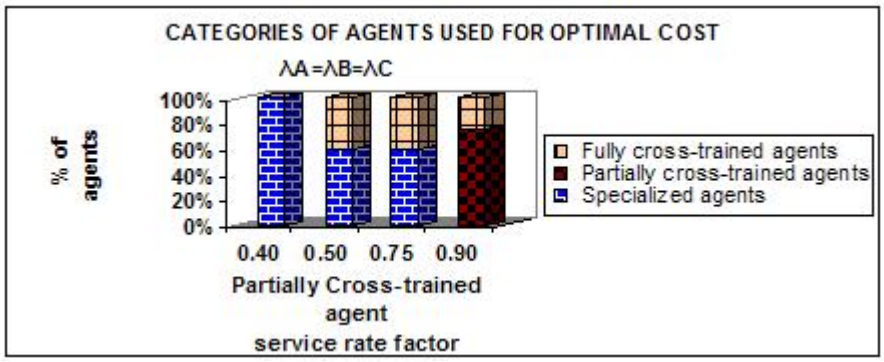


Figure 5.12: Optimal Agent Mix: $\lambda_A = \lambda_B = \lambda_C$, $S_1 = S_2 = S_3$

are involved in the agent combination that yields the best system performance. As the service rate factor increases, the number of cross-trained agents required for optimal cost increases.

5.6 Three-Customer Category: Different Arrival Rates, Varying Service Rates and Equal Agent Cost

As in the other cases and irrespective of the service rate factor being considered, the system cost decreases with an increase in the number of agents employed until the optimum point is attained. Thereafter, any additional agent used results in a higher system cost. Similarly, the optimal cost for lower service rate factors is generally

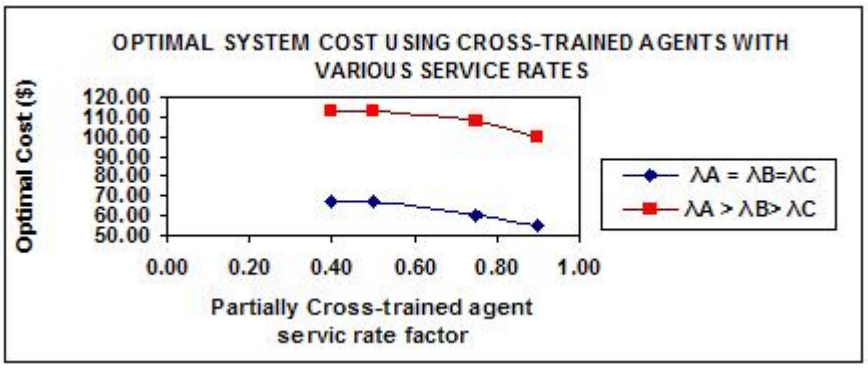


Figure 5.13: Optimal System Cost: $S_1 = S_2 = S_3$

higher than the optimal cost for higher service rate factors. For example, as shown in figures 5.14 and 5.13, when the service rate factor is 0.4 the optimal system cost is 113.00 dollars, but when a higher service rate factor of 0.9 is used this cost reduces to 99.90 dollars.

The service rate factor also affects the way specialized, partially cross-trained and fully cross-trained agents are allocated to yield the optimal system performance. Consider figure 5.15. When the service rate factor is 0.4 or 0.5, the use of only specialized agents gives the best system performance and no partially or fully cross-trained agent is utilized. The level of each of the categories of the specialized agents used is such that $n_A \geq n_B \geq n_C$, where n_A is the number of specialized A-agents utilized. As the factor increases, the level of specialized agents used decreases and there is a tendency of requiring more partially cross-trained or fully cross-trained agents for optimality. An important observation which follows a pattern from the two customer stream case is that there is no case when there is no specialized agent used, especially the specialized A-agents. This can be explained from the fact that A-customers have the highest arrival rate among the three streams of customers. A particular case of interest, though, is when the factor is 0.75. Here, all the three categories of agent, specialized, partially cross-trained and fully cross-trained are utilized. The allocation of the specialized agents is such that $n_A \geq n_B \geq n_C$. The bivalent agents are utilized at a level such that $n_{AB} \geq n_{AC} \geq n_{BC}$ where n_{AB} is the number of A-B-agents used. At a service rate factor is 0.9, it is strange to note that no fully cross-trained agent should be used for optimal system cost. The explanation

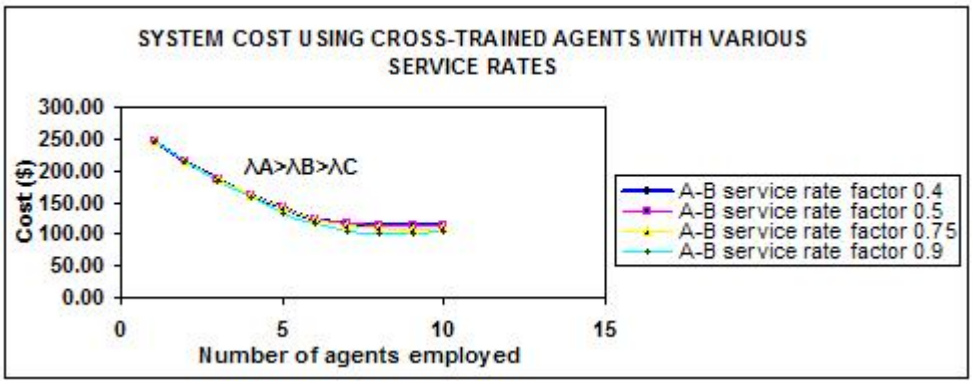


Figure 5.14: Agent Utilization and System Cost: $\lambda_A > \lambda_B > \lambda_C$, $S_1 = S_2 = S_3$

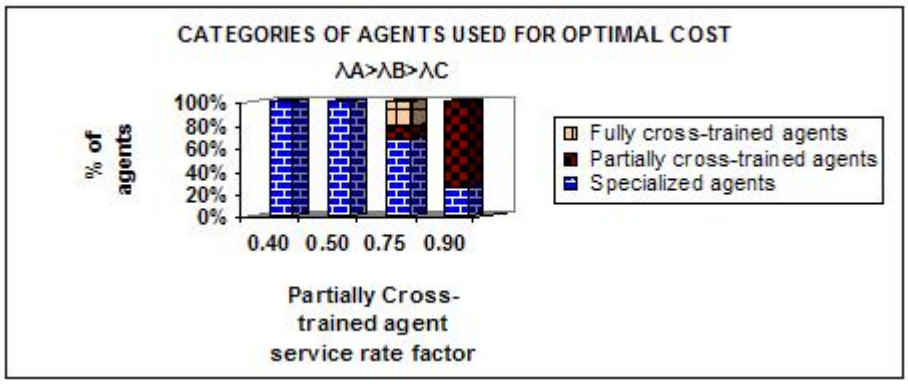


Figure 5.15: Optimal Agent Mix: $\lambda_A > \lambda_B > \lambda_C$, $S_1 = S_2 = S_3$

we can give here is that the preference of bivalent agent for service over a trivalent agent by a customer supersedes the effect of the flexibility of a trivalent agent given their service rates in this circumstance. Therefore, it is in this situation that the use of fully cross-trained agents in the three-customer category makes no sense. Instead, it is sufficient to utilize partially cross-trained agents in their place.

5.7 Three-Customer Category: Equal Arrival Rates, Equal Service Rates and Varying Agent Cost

We also define the partially cross-trained agent cost factor as the ratio of the cost of a partially cross-trained agent A-B per time unit to the cost of a specialized agent-A per time unit, or S_2/S_1. This ratio is greater or equal to one. If it is equal to one

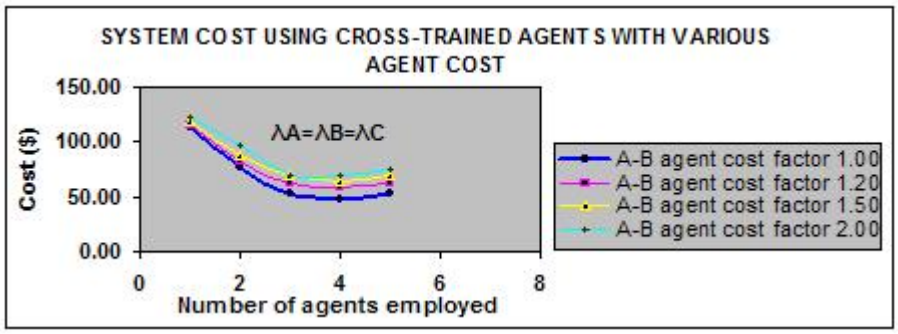


Figure 5.16: Agent Utilization and System Cost: $\lambda_A = \lambda_B = \lambda_C$, $\mu_1 = \mu_2 = \mu_3$

it means that the cost of a bivalent agent per time unit is equal to the cost of a univalent agent per time unit. A ratio greater than one means that the bivalent agent is more expensive than the univalent agent. For the trivalent agents we choose a cost per unit time of $S_3 = 1.25S_2$. A cost factor of 1.25 is reasonably low enough for a fully cross-trained agent to enhance their flexibility and make them competitive with their bivalent counterparts.

As with all the cases, given a partially cross-trained agent cost factor, the system cost decreases as the system capacity increases until the optimum cost is reached. After this point, the introduction of more agents into the system results in an increase in the system cost. As the agent cost factor increases, the optimal cost increases. For example, for an agent cost factor of 1.20 the optimal system cost is 59.10 dollars, while the optimal system cost increases to 64.80 dollars for a an agent cost of 1.50. This is depicted in figures 5.16 and 5.18. We also find that as the ratio increases, the level of cross-trained agents employed for optimal system performance decreases and the level of specialized agents utilized increases as shown in figure 5.17. When the ratio is one, only fully cross-trained agents are used to obtain the minimum cost. It is at this ratio that the flexibility of the trivalent agents is most obvious in the three-customer stream model. At the ratio of 2.0, no cross-trained agents are used. Again, because $\lambda_A = \lambda_B = \lambda_C$, any time specialized agents are used in the agent mix, the best performance occurs when the allocation is kept at $n_A = n_B = n_C$ if the number of specialized agent used is three or a multiple of three, and close to

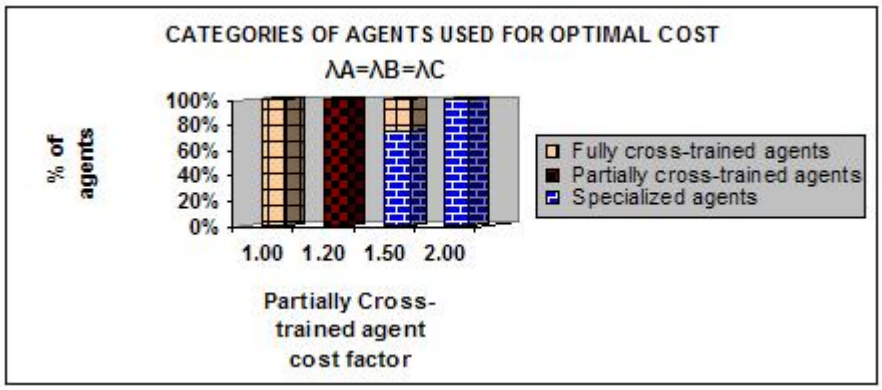


Figure 5.17: Optimal Agent Mix: $\lambda_A = \lambda_B = \lambda_C$, $\mu_1 = \mu_2 = \mu_3$

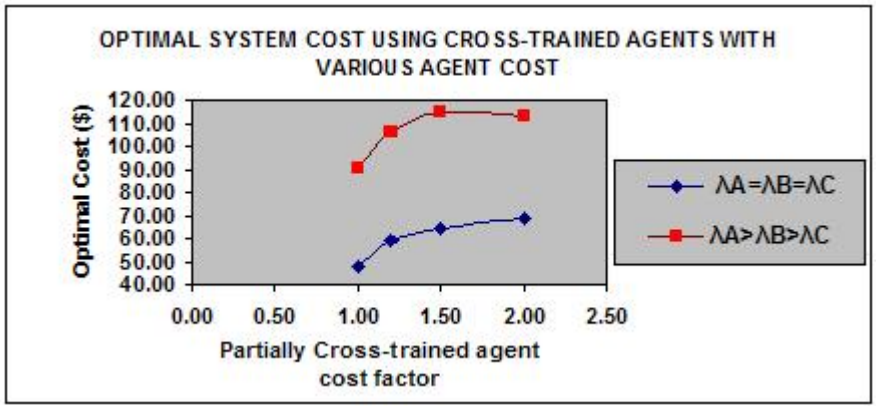


Figure 5.18: Optimal System Cost: $\mu_1 = \mu_2 = \mu_3$

$n_A = n_B = n_C$ otherwise. The same reason explains the need to keep the level of the partially cross-trained agents close to $n_{AB} = n_{AC} = n_{BC}$ when they are used to achieve the optimal system cost.

5.8 Three-Customer Category: Different Arrival Rates, Varying Agent Cost and Equal Service Rates

When the arrival rates of the three streams of customers are different, the system cost also reduces as more agents are employed until the optimal point after which the use of any more agents results in an increase in cost. The optimal cost is different for each partially cross-trained agent cost factor considered. It increases as the cost

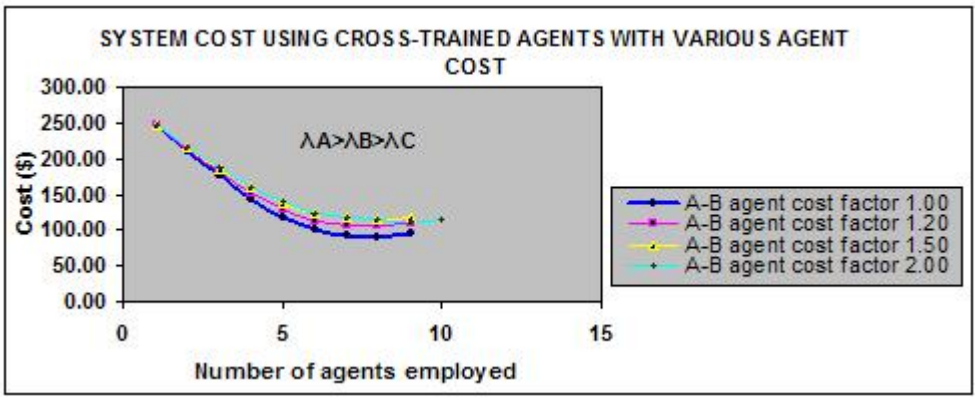


Figure 5.19: Agent Utilization and System Cost: $\lambda_A > \lambda_B > \lambda_C$, $\mu_1 = \mu_2 = \mu_3$

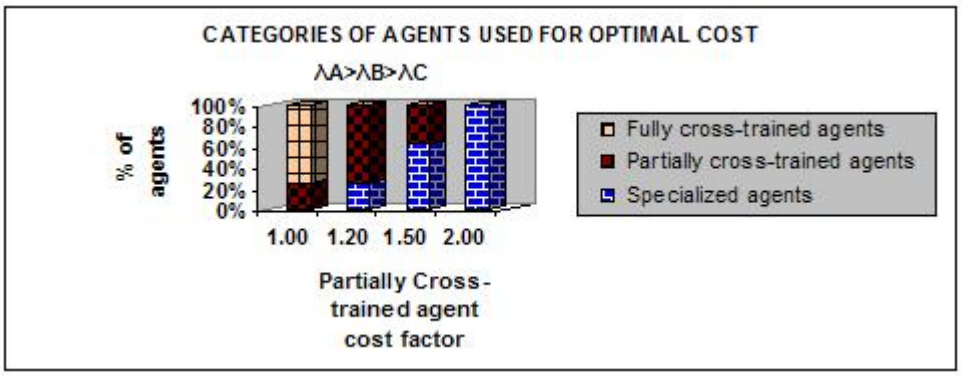


Figure 5.20: Optimal Agent Mix: $\lambda_A > \lambda_B > \lambda_C$, $\mu_1 = \mu_2 = \mu_3$

factor increases. For example, when the cost factor is 1.20, the optimal cost is 106.00 dollars, and when the cost factor is 1.5, the optimal cost increases to 115.00 dollars as can be seen in Figures 5.18 and 5.19.

In terms of the agent distribution at optimal system cost, we find that the number of bivalent and trivalent agents decrease as the agent cost factor increases as shown in Figure 5.20. Conversely, the number of specialized agents used increases with an increase in the cost factor. At a cost factor of 2.00, hundred percent of all agents used are specialized. At both cost factors of 1.20 and 1.50 partially cross-trained servers instead of fully cross-trained servers are used together with specialized servers for optimum performance. At any point when they are used for best performance, the allocation of univalent servers follows the scheme: $n_A \geq n_B \geq n_C$. The bivalent agents are allocated at a level such that $n_{AB} \geq n_{AC} \geq n_{BC}$ as shown in figure 5.20.

Chapter 6

Summary, Conclusion and Future Work

We present the summary of this thesis and its conclusion in this chapter.

6.1 Summary

The general goal of this research is to estimate the staffing levels in queueing systems with specialized and flexible agents. It focuses on finding the combination of these agents that results in an efficient system performance and optimal cost.

Chapter 1 focuses on the introduction to queueing systems and agent cross-training. Chapter 2 contains a literature review on contact centers, agent cross-training and finding the right number of servers. The two queueing system models that are studied are presented in chapter 3. One model has two streams of customers and the other has three streams. Chapter 3 also contains a brief description of Möbius and Eqsp tools, and uses both tools to solve a simple sample M/M/c/K queueing problem. The general framework is built for the three-category customer model, since it is easy to modify the framework to study the two-stream by assigning zero to the arrival rate of one of the three streams. By reducing the framework further to a one-category customer M/M/c/K model and comparing the results obtained from its use, we test the validity of the tools and our model framework in computing the necessary system performance measures.

In chapter 4, we itemize the questions that the thesis seeks to answer and plan the numerical experiments that are conducted to answer those questions. The numerical experiments to obtain the system performance measures of the queueing systems are done in Möbius. The system performance measures include loss probabilities and

the mean number of customers in the queues.

This thesis introduces two parameters which we called service rate factor and agent cost factor. The service rate factor compares the service rate of a cross-trained agent with the service rate of a specialized agent. On the other hand, the agent cost factor compares the cost of a cross-trained agent with the cost of a specialized agent. By varying these factors, we explore the extent to which specialized agents and flexible agents are mixed in achieving optimal system cost.

Chapter 5 deals with the analysis and evaluation of the results obtained from the experiments and attempts to answer the research questions posed in chapter 4.

6.2 Conclusion

Finding the right number of servers as well as the best ways to combine these servers when some of them are multi-skilled continues to be of importance in the study of queueing systems. The results from this thesis lead us to the following conclusions:

The cross-trained agent service rate factor and cross-trained agent cost factor are necessary in determining whether cross-trained agents are used to generate optimal system performance. They give us an idea of how fast or slow and how costly the cross-trained agents are compared with their specialized counterparts. The higher their service rate factors and the lower their cost factors, the more they are involved in achieving the best system cost. Instead of using an expensive and slow cross-trained server even though there is an advantage of flexibility, a less expensive and fast specialized server which could give a better system performance should be utilized. The advantages derived from cross-training are undermined if the cross-trained servers are too slow and/or too expensive.

In the two-category customer model with equal arrival rates, an attempt should be made to keep the number of the specialized agents for each category equal in order to get the optimum system cost. When the arrival rates are distinct, the mix should be such that the stream with the higher arrival rate is allocated more specialized agents.

In the three-category customer model with equal arrival rates, the best system cost is achieved by attempting to allocate equal number of specialized agents for each of the three categories. When they are used, the partially cross-trained agents should also be allocated equally such that $n_{AB} = n_{AC} = n_{BC}$. When their arrival rates are different with $\lambda_A > \lambda_B > \lambda_C$, the allocations which aim at $n_A \geq n_B \geq n_C$ and $n_{AB} \geq n_{AC} \geq n_{BC}$ yield the best solution.

6.3 Future Work

There are aspects of these thesis that need to be studied in the future. In our study, the service times are agent-dependent. One can investigate what happens when the customers have different service time requirements. This will be challenging in Möbius which, at present, supports only homogeneous tokens and does not have a way of differentiating one token from the other at a 'place'. There should be a way of capturing when a multivalent server is serving a customer from a particular category in a customer-dependent service time model.

Establishing staffing levels in queueing systems that have specialized and multi-skilled servers where priority is given to certain customers over others will be interesting, as well. Future work can also be done to extend the application to larger service centers which may have higher-than-three customer category models. Approximations to get the analytical solutions of these types of queueing models can also be studied.

Finally, sensitivity analysis of the waiting time cost and other costs on the optimal decisions can be investigated.

References

[1] Allen, A. O., (1990): *Probability, Statistics, and Queueing Theory with Computer Science Applications*, San Diego. Academic Press, Inc.

[2] Andrews, B., Parsons, H., (1993): Establishing Telephone-Agent Staffing Levels through Economic Optimization, Interfaces, Vol. 23(2), 14-20

[3] Bevilacqua Masi, D. M., Fischer, M. J., Harris, C. M., (2001): Computation of Steady-State Probabilities for Resource-Sharing Call-Center Queueing Systems, Stochastic Models, Vol. 17(2), 191-214

[4] Brockmeyer, E., Halstrom, H. L., Jensen, A., (1960): *The life and works of A K Erlang*, Acta Polytech. Scandinav. 287

[5] Chevalier, P., Tabordon, N., (2003): Overflow Analysis and Cross-trained Servers, Int. J. Production Economics, Vol. 85, 47-60

[6] Foulds, L. R., (1981): *Optimization Techniques: An Introduction*, New York. Springer-Verlag

[7] Grassmann, W. K., (1988): Finding the Right Number of Servers in Real-World Queuing Systems, Interfaces, Vol. 18, 94-104

[8] Grassmann, W. K., (2000): *Computational Probability*, Norwell, Massachusetts. Kluwer Academic Publishers

[9] Grassmann, W. K., (2005): Eqsp: A Software for Calculating the Equilibrium Solutions of Systems. Available from http://www.cs.usask.ca/classes496/t2/eqsp.c (accessed June 8, 2006)

[10] Groebner, D. F., Shannon, P. W., (1992): *Introduction to Management Science*, New York, New York. Macmillan Publishing Company

[11] Halfin, S., Whitt, W., (1981): Heavy Traffic Limits of Queues with Many Exponential Servers, Operations Research, Vol. 29, 567-588

[12] Linder, R. W., (1969): The Development of Manpower and Facilities Planning Methods for Airline Telephone Reservation Offices, Operational Research Quarterly, Vol. 20, 3-21

[13] Mandelbaum, A., (2004). Call centers (centres): Research bibliography with abstracts. Electronically available from http://ie.technion.ac.il/serveng/References/ccbib.pdf (accessed April 20, 2006).

[14] Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., (1989): *Numerical Recipes: The Art of Scientific Computing*, Cambridge, New York. Cambridge University Press

[15] Sanders, W.H., (2006). Möbius: A Model-based environment for validation of system reliability, availability, security and performance. Available from http://www.mobius.uiuc.edu/papers.html (accessed June 11, 2006)

[16] Sauer, T. D., (2006): *Numerical Analysis*, Boston. Pearson Addison Wesley

[17] Shumsky, R. A., (2004): Approximation and Analysis of a Call Center with Flexible and Specialized Servers, OR Spectrum, Vol. 26, 307-330

[18] Smith, D. K., (2002): Calculation of Steady-State Probabilities of M/M Queues: Further Approaches, Journal of Applied Mathematics and Decision Sciences, Vol. 6(1), 4350

[19] Stanford, D. A., Grassmann, W. K., (1993): Bilingual Server System: A Queueing Model Featuring Fully and Partially Qualified Servers, INFOR, Vol. 31, 261-277

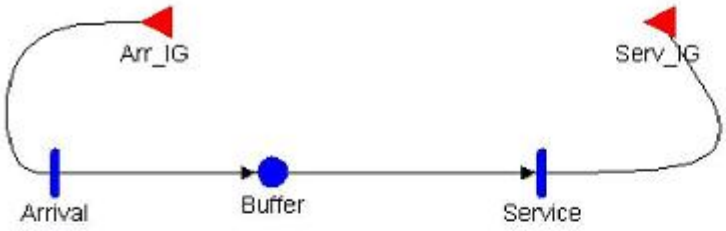
[20] Stanford, D. A., Grassmann, W. K., (2000): Bilingual Server Call Centres, Fields Institute Communication, Vol. 28, 31-47

[21] Tekin, E., Hopp, W. J., Van Oyen, M. P., (2009): Pooling Strategies for Call Center Agent Cross-training, IIE Transactions, Vol.41, 546-561

Appendix A

An M/M/c/K Solution Documentation in Möbius

Model: Queue



Bucket Attributes:

Place Names	Initial Markings
Buffer	0

Timed Activity:	Arrival
Exponential Distribution Parameters	**Rate** `ArrivalRate`
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	Service

Exponential Distribution Parameters	**Rate**
	`double myrate;`
	`if(Buffer->Mark()>num_server)`
	`{`
	`myrate=num_server*ServiceRate;`
	`}`
	`else`
	`{`
	`myrate=Buffer->Mark()*ServiceRate;`
	`}`
	`return myrate;`

Activation Predicate	(none)
Reactivation Predicate	(none)

Input Gate:	Arr_IG
Predicate	`Buffer->Mark() < BUF_SZ`

Function	`// This input gate enables activity Arrive. A token will be placed in Buffer.`
	`// We do not need to do anything for this input function. So, just specify`
	`// empty statement with semicolon below.`
	`//Buffer->Mark()++`
	`;`

Input Gate:	Serv_IG
Predicate	`Buffer->Mark() >0`

Function	`// This input gate enables activity Service. A token will be removed from Buffer.`
	`// We do not need to do anything for this input function. So, just specify`
	`// empty statement with semicolon below.`
	`//Buffer->Mark()--`
	`;`

Performance Variable Model: Queue_PV		
Top Level Model Information	Child Model Name	Queue
	Model Type	SAN Model

Performance Variable : Occupancy			
Affecting Models	Queue		
Impulse Functions			

Reward Function	*(Reward is over all Available Models)*
	`// Mathematically, this can be computed by rho/ (1 - rho).`
	`return (Queue->Buffer->Mark());`		

Simulator Statistics	Type	Steady State	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Initial Transient	0.0
		Batch Size	1.0
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Performance Variable : Utilization		
Affecting Models	Queue	
Impulse Functions		

Reward Function	*(Reward is over all Available Models)*
	`// When this activity fires and if there are jobs in the queue, then the queue is`
	`// being utilized (so return 1 to indicate this). When this activity fires and there`
	`// are no jobs in the queue, then the queue is not being utilized. Averaging this`
	`// reward variable over time gives us the utilization of the queue.`
	`// // Mathematically, this can be computed by rho = lambda / mu.`
	`if (Queue->Buffer->Mark() > num_server)`
	`{`
	`return (1.0);`
	`}`
	`else`
	`{`
	`return (Queue->Buffer->Mark()*1.0/num_server);`
	`}`	

	Type	Steady State
		Estimate Mean
		Include Lower Bound on Interval Estimate

Simulator Statistics	Options	Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Initial Transient	0.0
		Batch Size	1.0
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Performance Variable : AvgResponseTime			
Affecting Models	Queue		
Impulse Functions			

Reward Function	*(Reward is over all Available Models)*
	`// The mean response time is computed by the (number of jobs in the queue plus`
	`// the arriving job) times (1 / the service rate).`
	`// Mathematically, this can be computed by 1 / (mu - lambda).`
	`return ((Queue->Buffer->Mark() + 1) * (1.0 / ServiceRate));`		

Simulator Statistics	Type	Steady State	
	Options	Estimate Mean	
		Estimate Interval	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Initial Transient	0.0
		Batch Size	1.0
	Intervals	Lower Bound on Interval Estimate	0.0
		Upper Bound on Interval Estimate	1.0
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Performance Variable : lossprobability		
Affecting Models	Queue	
Impulse Functions		

Reward Function	*(Reward is over all Available Models)*
	`if (Queue->Buffer->Mark() >=BUF_SZ)`
	`{`
	`return (1.0);`
	`}`
	`else`
	`{`
	`return (0.0);`
	`}`	

Simulator Statistics	Type	Steady State
	Options	Estimate Mean
		Include Lower Bound on Interval Estimate
		Include Upper Bound on Interval Estimate
		Estimate out of Range Probabilities
		Confidence Level is Relative

	Parameters	Initial Transient	0.0
		Batch Size	1.0
	Confidence	Confidence Level	0.95
		Confidence Interval	0.1

Range Study Variable Assignments for Study *Queue_PR* in Project *M_M_C_K* :

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
ArrivalRate	double	Fixed	4.0	-	-	-	-
BUF_SZ	short	Fixed	5	-	-	-	-
ServiceRate	double	Fixed	3.0	-	-	-	-
num_server	short	Incremental	[Incremental Range,1]	1	Additive	-	-

HTML generated by Möbius Documentor

Figure A.1: An M/M/c/K Atomic Model Solution Documentation

Mobius sor Version 1.8.0,
Copyright (c) William H. Sanders and Univ. of Illinois 1994-2005

****
******
Mobius ITERATIVE SOR STEADY STATE SOLVER

ITERATIVE SOR STEADY STATE SOLVER RESULTS
****
******

Started: Sat Apr 07 18:00:17 2007

Project name: M_M_C_K
Study name: Queue_PR
State space name: Queue_SSG
Experiment name: Experiment_1

Global variable settings for this experiment:
ArrivalRate = 4.0
BUF_SZ = 5
ServiceRate = 3.0
num_server = 3

Verbosity: 0
Accuracy: 1.000000e-009
Weight: 1.000000
Max iterations: 300000
Matrix diagonalization off.

Computation Time (seconds): 0.000000
User Time (seconds): 0.000000
System Time (seconds): 0.000000
Number of states in process: 6
Number of non-zero elements: 16
Solution converged in 31 iterations.
Maximum difference: 7.190576e-010

```
Computing the performance variables
************************************************************************
******
Performance variable :  Occupancy
Mean              :  1.392133e+000
Variance          :  1.473235e+000
Plot files   (pdf) :  Experiment_1.sor.Occupancy.pdf.splot
             (cdf) :  Experiment_1.sor.Occupancy.PDF.splot
************************************************************************
******
Performance variable :  Utilization
Mean              :  4.354845e-001
Variance          :  1.182637e-001
Plot files   (pdf) :  Experiment_1.sor.Utilization.pdf.splot
             (cdf) :  Experiment_1.sor.Utilization.PDF.splot
************************************************************************
******
Performance variable :  AvgResponseTime
Mean              :  7.973775e-001
Variance          :  1.636927e-001
Plot files   (pdf) :  Experiment_1.sor.AvgResponseTime.pdf.splot
             (cdf) :  Experiment_1.sor.AvgResponseTime.PDF.splot
************************************************************************
******
Performance variable :  lossprobability
Mean              :  2.015986e-002
Variance          :  1.975344e-002
Plot files   (pdf) :  Experiment_1.sor.lossprobability.pdf.splot
             (cdf) :  Experiment_1.sor.lossprobability.PDF.splot
Finished: Sat Apr 07 18:00:17 2007
```

Figure A.2: An M/M/c/K Performance Variables Solution Documentation