Performance Prediction for Designing Shared Services

by

Yu Liu

A Thesis Submitted to Saint Mary's University, Halifax, Nova Scotia,
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Applied Science

April 23, 2014, Halifax, Nova Scotia

Approved: Dr. Hai Wang
Supervisor
Department of Finance, Computing and Information Systems and Management Science

Approved: Dr. Yinglei Wang
External Examiner
School of Business
Acadia University

Approved: Dr. Michael Zhang
Supervisory Committee Member
Department of Finance, Computing and Information Systems and Management Science

Approved: Dr. Genlou Sun
Supervisory Committee Member
Department of Biology

Approved: Dr. Stavros Konstantinidis
Graduate Studies Representative

Date: April 23, 2014

Table of Content

Performance Prediction for Designing Shared Services

By Yu Liu

Abstract

Since early 1980s, *shared services* have been utilized by public and private organizations with the purpose of reducing the administrative cost. Currently, shared services evolve into an efficient and flexible tool in optimizing resources and capitals, raising service qualities, promoting strategic growth and generating greater profits for both public and private organizations. One important aspect of design and implementation of shared services is to ensure the quality services delivered by a shared service center. This thesis presents a new family of approximate Mean Value Analysis algorithm for solving multi-class product-form queuing network models. The proposed algorithms are capable of quickly and accurately predicting the average completion time of different types of tasks to be delivered by a shared service center. The computational and numerical properties of the proposed algorithms are analyzed. This thesis demonstrates the usefulness and effectiveness of the proposed algorithms for facilitating the design and implementation of shared services.

April 23, 2014

Chapter 1. Introduction

In today's world, business competition is unprecedentedly fierce. As the result, many organizations attempt to streamline the non-core business processes and reduce the corresponding costs so that they can focus on the core business processes. Since early 1980s, *shared services* have been utilized by private companies with the purpose of reducing the administrative cost (Ask *et al.* 2008, Borman 2010, Janssen & Joha 2006, Kakabadse & Kakabadse 2000, Kamal 2012, Lindvall & Iveroth 2011, McDowell 2011, Minnaar 2013, Niehaves & Krause 2010, Ulbrich 1995, Ulbrich 2010). Currently, shared services evolve into an efficient and flexible tool in optimizing resources and capitals, raising service qualities, promoting strategic growth and generating greater profits for both public and private organizations. Accordingly, the prediction and evaluation of the appropriateness and flexibility of the shared services is highly demanded.

There are two types of shared services, *internal* and *external*, that have been widely spread in many government and business organizations (Gulati & Singh 1998, Kakabadse & Kakabadse 2000, Kamal 2012, Lindvall & Iveroth 2011). For *internal shared*

services, the organization restructures its existing business processes, and common non-core business functions are consolidated and shared by different operational units within the organization. For *external shared services*, multiple partner organizations form a strategic alliance, and common non-core business functions are standardized and consolidated across multiple partner organizations. External shared services have been adopted by many government and business organizations as an alternative solution to *outsourcing* (Wang & Wang 2007). For both types of shared services, the services are delivered by shared service centers. Shared service centers are the core component of shared services. Shared service centers attempt to

1. Reduce the operational costs of non-core business functions and processes of the organization.
2. Enable an organization to increase its focus on core business functions and processes.
3. Increase information and knowledge sharing, either within the organization (for internal shared services) or across multiple organizations (for external shared services).

As shared service centers in nature are complicated systems, it is important to estimate and predict the quality of services provided by shared service centers during the design

stage (Rolia *et al.* 2006, Wang 2007, Wang & Wang 2008, Wang *et al.* 2014). For a shared service center, there are multiple types of tasks from different sources with different requirements on the completion time. It has been suggested that *multi-class product-form queuing network models* are useful and applicable in predicting the performance of complex systems like shared service centers (Rolia *et al.* 2006, Wang 2007, Wang & Wang 2008, Wang *et al.* 2014).

In this thesis, we propose a new family of algorithms that are capable of quickly and approximately solving multi-class product-form queuing network models. These new algorithms can be used to facilitate the design and implementation of shared service centers by predicting the completion time of different types of tasks and examining whether all requirements from different sources are fulfilled. We also investigate the properties of the proposed algorithms, and demonstrate their usefulness to the design of shared service centers.

The remaining four chapters are organized as follows. Chapter 2 is divided into two major parts. The first part discusses the background of internal and external shared services, and the second part discusses the background of multi-class product-form queuing

network models and the algorithms for solving these models. Chapter 3 presents a new family of algorithms for solving multi-class product-form queuing network models. Chapter 4 presents an experimental analysis of the computational and numerical properties of the proposed algorithms. Finally, Chapter 5 presents our conclusions and discusses the future work.

Chapter 2. Background

2.1. Shared Services

2.1.1. Features of shared services

Shared services are often defined as a shared service center delivering services to different operational units within an organization or even to different organizations in a standardized and centralized way. The delivery of shared services, which is ensured through business processes, provides great strategic opportunities for cost saving as well as for information and knowledge sharing (Davenport *et al*., 2004; Davenport and Short, 1990).

There are two types of shared services, internal and external. For internal shared services, the shared service center delivers services to different operational units within an organization. Large government and business organizations often have many operational units. Each operational unit not only focuses on its core business functions, but also supports necessary non-core business functions. Internal shared services attempt to reduce the operational cost for such large organizations. To implement internal shared services, the organization restructures its existing business processes, and common non-core business functions are consolidated and shared by different operational

units within the entire organization. Generally, non-core business functions, such as information system services and human resource management, are designated line of business processes for internal shared services.

For external shared services, multiple partner organizations form a strategic alliance and establish a shared service center to deliver services to these partner organizations. The shared service center is capable of reducing operational cost and increasing information and knowledge sharing among the partner organizations. External shared services usually standardize and consolidate common non-core business functions across multiple partner organizations so that the partner organizations can increase their focus on the core business functions. The information and knowledge sharing aspect of external shared services enable the partner organizations to gain extra competitive advantages over other competitors in addition to cost reduction. External shared services have been widely implemented by many government and healthcare organizations as an alternative solution to outsourcing (Kamal 2012, McDowell 2011, Niehaves & Krause 2010, Ulbrich 2010).

With the wide adoption of both internal and external shared services, the range of shared services have also been expanding, which include accounting and financial services,

information system services, customer relationship management, human resource management.

2.1.2. Major goals of shared services

The major goals of internal shared services are:

1. To reduce the operational costs of non-core business functions and processes through business process re-engineering (BPR) and business process optimization (BPO).
2. To Increase focus on core business functions and processes.
3. To increase information and knowledge sharing among different operational units within the organization.
4. To improve the quality of the services delivered by the shared service centers through business process re-engineering (BPR) and business process optimization (BPO) as well as information and knowledge sharing.

The goals of external shared services are similar:

1. To establish a mutual beneficial and long-term strategic relationship with other partner organizations to share the governance, costs and risks of the shared service centers.

2. To reduce the operational cost of non-core business processes through standardizing and consolidating the non-core business processes across multiple partner organizations.
3. To increase information and knowledge sharing among the partner organizations to enable these organizations to gain extra competitive advantages over other competitors in addition to cost reduction.
4. To improve the quality of the services delivered by the shared service centers through business process re-engineering (BPR) and business process optimization (BPO) as well as information and knowledge sharing.

2.1.3. Distinctions between external shared services and outsourcing

External shared services are often mistreated as *outsourcing*. As a matter of fact, there lies a fundamental difference between them. For external shared services, the shared service center is formed and governed by the partner organizations (Gulati & Singh 1998, Kakabadse & Kakabadse 2000). The partner organizations establish a long-term strategic alliance, and share the governance, cost and risks of the shared service center.

For outsourcing, the relationship between the outsourced organization and the third-party

service provider is defined by a bilateral contract, which involves an exchange of services and payments (King 2006). Outsourcing provides the outsourced organization a great financial flexibility. The outsourced organization can pay for only the services it needs, at the time it really needs them. However, outsourcing often associates with many potential risks, such as lack of transparency, loss of control of data and business knowledge, failure to maintain the service quality.

External shared services can be viewed as an alternative solution to outsourcing. The cost saving of external shared services is long-term and stable, while that of outsourcing is often short-term. An important advantage of external shared services over outsourcing is that external shared services increase the information and knowledge sharing among multiple partner organizations while the organizations still have control over the information and knowledge to be shared. The information and knowledge sharing aspect of external shared services gives the partner organizations an extra competitive advantage over other competitors.

2.1.4. Design and implementation of shared services

Both internal and external shared services require organizational structure changes.

During the design and implementation of internal or external shared services, business process reengineering (BPR) and business process optimization (BPO) are often used to re-examine the existing business processes, and identify the business processes and business functions for shared service centers (Ulbrich 2006, Wang & Wang 2007, Wang & Wang 2014). In fact, as previous research indicates, a new organizational structure is highly needed to ensure the success of shared services (Gulati & Singh 1998, Kakabadse & Kakabadse 2000). With the assistance of shared services, the organizations are able to shift non-core business process to the shared service center, and centralize their efforts and focus on the core business processes to win the fierce competitions. One important aspect of successful design and implementation of internal and external shared services is to ensure the quality of services delivered by the shared service center.

For internal shared services, the shared service center delivers services to different operational units within an organization which may have different requirements about the levels and qualities of services. For external shared services, the shared service center delivers services to different partner organizations which may have different requirements about the levels and qualities of services. Hence, regardless of the types of shared services, the services delivered by the shared service center generally have different

requirements about the levels and qualities of services. It is necessary to ensure all these requirements are fulfilled when designing and implementing shared services. In the next subsection, we discuss an analytical tool for predicting and evaluating the performance of shared service centers.

2.2.Queuing Network Modelling for Shared Services

2.2.1. Queuing network modelling

Modelling a real-life system by nature is an abstraction. Queuing network models are simple models at a high level of abstraction. It avoids the unnecessary figures and concentrates on the core details. Performance measures such as system throughput and response time can be defined, parameterized and evaluated. Queuing network models have been recognized as a powerful and versatile tool in the design and implementation of computer and communication systems (Buzen 1973, Cremonesi *et al.* 2002, Pattipati *et al.* 1990).

Since 1970s, the rapid evolvement and increasing complexity of the computer systems have given rise to the increasing need for analytical tools to predict and evaluate the performance and behavior of these systems. Among various analytical tools, queuing

network models have been widely used for performance modeling of computer systems due to a favorable balance between their accuracies and efficiencies (Cremonesi *et al.* 2002, Pattipati *et al.* 1990, Rolia *et al.* 2006). In this thesis, we will use queuing network models to predict the performance of shared service centers during the design of shared services.

A queuing network model consists of a collection of customers (users) and service centers (servers). The service centers represent the resources for providing services to customers. The customers' competition for the resource service can be interpreted as queuing to the service center. After being served at a service center, the user will depart from it and may join another queue.

A solution of a queuing network model is a set of performance measures such as system throughput and customer response time. While there is no exact solution available for the general class of queuing network models, exact solutions can be computed for a special type of queuing network models, called *multi-class product-form queuing network models*. Many other types of queuing network models can be accurately approximated by larger and more complex multi-class product-form queuing network models (Bolch *et*

al. 2006).

2.2.2. Multi-class product-form queuing network models

In 1975, Baskett, Chandy, Muntz and Palacios presented the multi-class product-form queuing network models involving multiple classes of customers (Baskett *et al.* 1975). Since then, multi-class product-form queuing network models have been widely used for performance prediction and evaluation of the complex computer systems.

A multi-class product-form queuing network model is a queuing network model that meets the following three conditions:

1. The queues in the queuing network model can hold an infinite number of customers.
2. The customer service time at a service center is non-deterministic, which means the actual service time of a customer follows a probability distribution. Customers in the same class share the same probability distribution for the service time at a particular service center.
3. Any resource service center in the queuing network model must belong to one of the following four types:

1) Type I: Service centers with only one server. Customers are served with the *First Come First Served scheduling*. Customer service time at the service center follows an *exponential distribution*, and different classes of customers all have this customer service time distribution.

2) Type II: Service centers with only one server. Customers are served with the *Processor Sharing scheduling* at the server center. Customer service time distribution can be an arbitrary distribution.

3) Type III: Service centers with only one server. Customers are served with the *Last Come First Served with Pre-emption scheduling*. Customer service time distribution can be an arbitrary distribution.

4) Type IV: Service centers with an infinite number of servers. Customers are served immediately without waiting. Customer service time distribution can be an arbitrary distribution.

The first three types of service centers, all customers are served solely by one server, and the customers need to wait in the queues for services. This is not the case of the last type of service centers, where there is no queuing and all customers are served immediately upon their arrival. The first three types of service centers represent a single resource required by different customers. The last type of service centers represents the infinity

resources that are always available. For all four types of service centers, the customer service time at a service center is non-deterministic, which means the actual service time of a customer follows a probability distribution.

Multi-class product-form queuing network models have be used to model shared service centers for both internal and external shared services (Rolia et al. 2006, Wang 2007, Wang & Wang 2008, Wang *et al.* 2014). For instance, a shared service center with two employees, A and B, offers house renting services to two different apartments, X and Y. Apartment X has two types of tasks. The first one requires A to offer house renting service, while the second one involves both A and B, with A offering house renting service first and B performing house cleaning next. Apartment Y has one type of tasks that require B to perform house cleaning service. Hence, there are three different types of tasks for this shared service center, two for apartment X (requiring A and A + B respectively), and one for apartment Y (requiring B). As shown in Figure 1, this shared service center can be modeled by a multi-class product-form queuing network with two service centers which represent two employees A and B respectively (service center A and service center B) and three classes of customers which represent the three different types of tasks respectively (class 1, class 2, and class 3). In this simple example, the nature of

the shared service center is not important, and it can be either internal or external. If both apartments X and Y belong to the same owner, then the shared service center in Figure 1 is internal. If apartments X and Y belong to different owners, then the shared service center in Figure 1 is external.

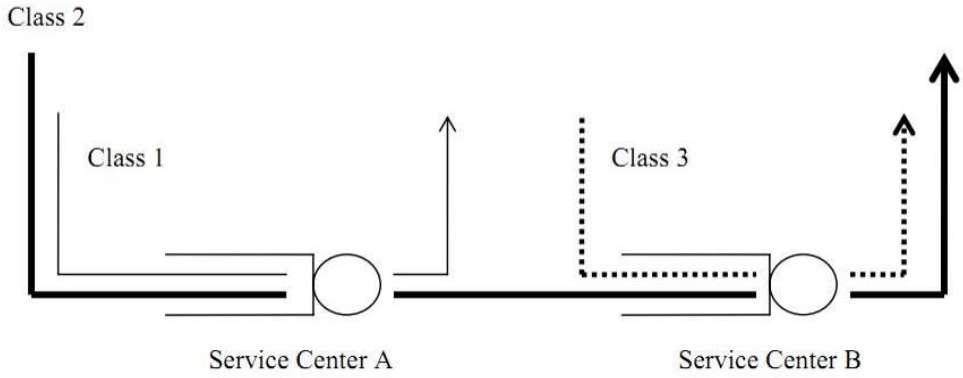

Figure 1. A Multi-Class Product-Form Queuing Network Model for the House Renting Shared Service Center

2.2.3. The MVA algorithm

One major reason of the popularity of multi-class product-form queuing network models is that their performance can be computed by various algorithms. Among these algorithms, the Mean Value Analysis (MVA) algorithm enjoys the widest popularity

(Reiser & Lavenberg 1980).

Suppose that a multi-class product-form queuing network model has C customer classes and M service centers. These M service centers can be any of Type I, II, III service centers described in Section 2.2.2. The customer classes are indexed as classes 1 through C, and the service centers are indexed as centers 1 through K. The customer population in the queuing network model is denoted by the vector $\mathbf{P} = [P_1, P_2, ..., P_C]$, where P_c is the number of customers of class c for c=1,2,...,C. The average service time of a class c customer at center k is denoted by $X_{c,k}$ for c=1,2,...,C, and k=1,2,...,K. The sum of the average service time of all Type IV service centers for class c is denoted by Y_c.

Given the customer population vector **P**, the performance measures of the multi-class product-form queuing network model are as follows:

- $R_{c,k}(\mathbf{P})$ = the average response time of a class c customer at center k.
- $R_c(\mathbf{P})$ = the average response time of a class c customer in the network.
- $Q_{c,k}(\mathbf{P})$ = the average queue length of class c at center k.
- $Q_k(\mathbf{P})$ = the average total queue length at center k.

Intuitively, $R_{c,k}(\mathbf{P})$ is the sum of the average service time and queuing waiting time of a class c customer at center k, and $R_c(\mathbf{P})$ is the sum of $R_{c,k}(\mathbf{P})$ for all centers in the queuing network model.

The Mean Value Analysis algorithm (Reiser and Lavenberg, 1980) involves repeated applications of four recursive equations:

$$R_{c,k}(\mathbf{p}) = X_{c,k} \cdot [1 + Q_k(\mathbf{p}\text{-}\mathbf{1}_c)] \qquad (1)$$

$$R_c(\mathbf{p}) = \sum_{k=1}^{K} R_{c,k}(\mathbf{p}) \qquad (2)$$

$$Q_{c,k}(\mathbf{p}) = \frac{\mathbf{p}_c \cdot R_{c,k}(\mathbf{p})}{Y_c + R_c(\mathbf{p})} \qquad (3)$$

$$Q_k(\mathbf{p}) = \sum_{c=1}^{C} Q_{c,k}(\mathbf{p}) \qquad (4)$$

with initial conditions $Q_k(\mathbf{0}) = 0$ for k=1,2,...,K, where $\mathbf{p} = [p_1, p_2, ..., p_C]$ is a population vector ranging from $\mathbf{0} = [0, 0, \ldots, 0]$ to $\mathbf{P}$, and $\mathbf{1}_c$ is a C-dimensional vector whose c^{th} element is one and whose other elements are zeroes. In Equation (1), $\mathbf{p}\text{-}\mathbf{1}_c$ is the population vector $\mathbf{p}$ with one class c customer removed. This recursive dependence indicates that the performance measures for one population can be computed from those for lower population levels.

The implementation of the Mean Value Analysis algorithm is as follows:

Algorithm Input:

C: the number of classes of customers

K: the total number of all Type I, II, III service centers

P: the customer population vector

Y: the vector of the sum of the average service time of all Type IV service centers for each customer class

X: the matrix for the average service time for each customer class and for each service center

Algorithm Output:

$R_{c,k}(\mathbf{P})$ = the average response time of a class c customer at center k

$R_c(\mathbf{P})$ = the average response time of a class c customer in the network

$Q_{c,k}(\mathbf{P})$ = the average queue length of class c at center k

$Q_k(\mathbf{P})$ = the average total queue length at center k

The MVA algorithm:

// initialization
FOR k = 1 TO K
    $Q_k(\mathbf{p}) = 0$

END FOR

FOR $p_1 = 0$ TO P_1
FOR $p_2 = 0$ TO P_2
……
FOR $p_C = 0$ TO P_C
p = the population vector [p_1, p_2, ..., p_C]
FOR c = 1 TO C
FOR k = 1 TO K
IF **p** = = **0** THEN
$R_{c,k}(\mathbf{p}) = 0$
$Q_{c,k}(\mathbf{p}) = 0$
ELSE
$R_{c,k}(\mathbf{p}) = X_{c,k} \cdot [1 + Q_k(\mathbf{p}\text{-}\mathbf{1}_c)]$
$$Q_{c,k}(\mathbf{p}) = \frac{\mathbf{p}_c \cdot R_{c,k}(\mathbf{p})}{Y_c + R_c(\mathbf{p})}$$
END IF
END FOR
END FOR
FOR k = 1 TO K
$$Q_k(\mathbf{p}) = \sum_{c=1}^{C} Q_{c,k}(\mathbf{p})$$
END FOR
FOR c = 1 TO C
$$R_c(\mathbf{p}) = \sum_{k=1}^{K} R_{c,k}(\mathbf{p})$$
END FOR
END FOR
……
END FOR
END FOR

2.2.4. Approximate MVA algorithms

The recursive Equation (1) cause both the space and time complexity of the MVA algorithm to be $\theta(\mathrm{KC}\prod_{c=1}^{C}(P_c+1))$ (Reiser & Lavenberg 1980). For the large models with C > 5 and K > 20, the computational time of the MVA algorithm will be prohibitively long, which means it is not practical to use it to compute the performance measures of the queuing models.

Under such circumstances, researchers turned to the approximate Mean Value Analysis (AMVA) algorithms (Bard 1979, Chandy & Neuse 1980, Eager & Sevcik 1984, Eager & Sevcik 1986, Cremonesi *et al.* 2002, Wang & Sevcik 2000, Pattipati *et al.* 1990, Wang *et al.* 2008). By using approximation instead of exact computation, the AMVA algorithms trade the solution accuracy for a lower computational complexity. The algorithm input and output of the AMVA algorithms are the same as those of the MVA algorithm. The only difference is that the MVA algorithm yields the exact solution while the AMVA algorithms yield an approximate solution with reduced computational time.

Among various AMVA algorithms, Proportional Estimation (PE) algorithm (Bard 1979) and the Linearizer algorithm (Chandy & Neuse 1980) are two most popular algorithms that have gained wide acceptance.

2.2.4. 1 The PE algorithm

The Proportional Estimation (PE) algorithm is based on the approximation

$$Q_k(\mathbf{P}\text{-}\mathbf{1}_c) = Q_k(\mathbf{P}) - Q_{c,k}(\mathbf{P}) / P_c \quad (5)$$

and it computes the average performance measures by solving the following system of nonlinear equations iteratively until convergence:

$$R_{c,k}(\mathbf{P}) = X_{c,k} \cdot [1 + Q_k(\mathbf{P}) - Q_{c,k}(\mathbf{P}) / P_c] \quad (6)$$

$$R_c(\mathbf{P}) = \sum_{k=1}^{K} R_{c,k}(P) \quad (7)$$

$$Q_{c,k}(\mathbf{P}) = \frac{P_c \cdot R_{c,k}(P)}{Y_c + R_c(P)} \quad (8)$$

$$Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P) \quad (9)$$

The Proportional Estimation algorithm is as follows:

Algorithm Input:

C: the number of classes of customers

K: the total number of all Type I, II, III service centers

P: the customer population vector

Y: the vector of the sum of the average service time of all Type IV service centers for each customer class

X: the matrix for the average service time for each customer class and for each service center

Algorithm Output:

$R_{c,k}(\mathbf{P})$ = the average response time of a class c customer at center k

$R_c(\mathbf{P})$ = the average response time of a class c customer in the network

$Q_{c,k}(\mathbf{P})$ = the average queue length of class c at center k

$Q_k(\mathbf{P})$ = the average total queue length at center k

The PE algorithm:

// initialization
FOR k = 1 TO K
    FOR c = 1 TO C
        $Q_{c,k}(\mathbf{P}) = P_c / K$
    END FOR
    $Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P)$
END FOR

// iterations
REPEAT
    FOR c = 1 TO C
        FOR k = 1 TO K
            $R_{c,k}(\mathbf{P}) = X_{c,k} \cdot [1 + Q_k(\mathbf{P}) - Q_{c,k}(\mathbf{P}) / P_c]$
        END FOR

$$R_c(\mathbf{P}) = \sum_{k=1}^{K} R_{c,k}(P)$$

END FOR

FOR c = 1 TO C

FOR k = 1 TO K

$$Q_{c,k}(\mathbf{P}) = \frac{P_c \cdot R_{c,k}(P)}{Y_c + R_c(P)}$$

END FOR

$$Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P)$$

END FOR

UNTIL $Q_{c,k}(\mathbf{P})$ ARE ALL STABLE (i.e., the changes in $Q_{c,k}(\mathbf{P})$ are small enough)

The PE algorithm is able to yield fairly accurate solutions. The errors of the approximate solutions are typically less than 10% with respect to the exact solution (Wang *et al.* 2008). The space complexity of the algorithm is O(KC), and the time complexity of the algorithm is O(KC) per iteration. In most cases, the Proportional Estimation algorithm converges quickly (Wang *et al.* 2008).

2.2.4.2. The Linearizer Algorithm

Compared with the PE algorithm, the solutions computed by the Linearizer algorithm, though at a comparatively higher computational cost, are much more accurate than those of the PE algorithms (Chandy & Neuse 1980). The Linearizer is based on the approximation

$$Q_k(\mathbf{P}\text{-}\mathbf{1}_c) = \sum_{i=1}^{C} (P_i - \omega_{i,c}) \cdot (Q_{i,k}(\mathbf{P}) / P_i) + \lambda_{c,k}(\mathbf{P}) \quad (10)$$

where

$$\omega_{i,c} = \begin{cases} 0 & for \quad i \neq c \\ 1 & for \quad i = c \end{cases}$$

and the $\lambda_{c,k}(\mathbf{P})$ term is an unknown error term. Theoretically, Equation (10) holds exactly when

$$\lambda_{c,k}(\mathbf{P}) = \sum_{i=1}^{C} (P_i - \omega_{i,c}) \cdot (Q_{i,k}(\mathbf{P}\text{-}\mathbf{1_c}) / (P_i - \omega_{i,c}) - Q_{i,k}(\mathbf{P}) / P_i)$$

The Linearizer algorithm approximates the $\lambda_{c,k}(\mathbf{P})$ term as

$$\lambda_{c,k}(\mathbf{P}) = \lambda_{c,k}(\mathbf{P}\text{-}\mathbf{1_j}) \quad (11)$$

for all j = 1, 2, … C.

The Linearizer algorithm employs the Core algorithm as a subroutine which involves solving the following system of nonlinear equations iteratively until convergence:

$$R_{c,k}(\mathbf{P}) = X_{c,k} \cdot [1 + \sum_{i=1}^{C} (P_i - \omega_{i,c}) \cdot (Q_{i,k}(\mathbf{P}) / P_i) + \lambda_{c,k}(\mathbf{P})] \quad (12)$$

$$R_c(\mathbf{P}) = \sum_{k=1}^{K} R_{c,k}(P) \quad (13)$$

$$Q_{c,k}(\mathbf{P}) = \frac{P_c \cdot R_{c,k}(P)}{Y_c + R_c(P)} \quad (14)$$

$$Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P) \quad (15)$$

The Core algorithm is similar to the PE algorithm. If the $\lambda_{c,k}(\mathbf{P})$ term in Equation (12) is zero, then the Core algorithm is exactly the same as PE algorithm. The Linearizer algorithm iteratively solves the Core algorithm at the original and reduced customer population levels, and estimates and updates the $\lambda_{c,k}(\mathbf{P})$ term using Equation (11) for the next round of iteration. Hence, the Linearizer algorithm involves two kinds of iterations, inner iterations and outer iterations. The iterations of the Core algorithm are called the inner iterations, and the iterations for estimating and updating the $\lambda_{c,k}(\mathbf{P})$ terms are referred to as the outer iterations. The time complexity of the inner iterations is $O(KC^2)$, and the space complexity of the inner iterations is O(KC). Each outer iteration involves O(C) executions of the inner iterations. The overall time complexity of the Linearizer algorithm is $O(KC^3)$ and the space complexity remains O(KC).

The Linearizer algorithm is as follows:

Algorithm Input:

C: the number of classes of customers

K: the total number of all Type I, II, III service centers

P: the customer population vector

Y: the vector of the sum of the average service time of all Type IV service centers for each customer class

X: the matrix for the average service time for each customer class and for each service center

Algorithm Output:

$R_{c,k}(\mathbf{P})$ = the average response time of a class c customer at center k

$R_c(\mathbf{P})$ = the average response time of a class c customer in the network

$Q_{c,k}(\mathbf{P})$ = the average queue length of class c at center k

$Q_k(\mathbf{P})$ = the average total queue length at center k

The Linearizer algorithm:

// initialization
FOR k = 1 TO K
    FOR c = 1 TO C
        $\lambda_{c,k}(\mathbf{P}) = 0$
    END FOR
END FOR

// iterations
REPEAT
    // solving the Core algorithm at the original population level
    $\forall c, k : Q_{c,k}(\mathbf{P})$ = Core (C, K, **P**, **Y, X,** $\boldsymbol{\lambda}$)

// solving the Core algorithm at the reduced population levels

FOR j=1TO C

$\forall c, k : Q_{c,k}(\mathbf{P\text{-}1_j}) = \text{Core}(C, K, \mathbf{P\text{-}1_j}, \mathbf{Y}, \mathbf{X}, \boldsymbol{\lambda})$

END FOR

// updating the λ matrix

FOR k = 1 TO K

FOR c = 1 TO C

$$\lambda_{c,k}(\mathbf{P}) = \sum_{i=1}^{C} (P_i - \omega_{i,c}) \cdot \left(Q_{i,k}(\mathbf{P\text{-}1_c}) / (P_i - \omega_{i,c}) - Q_{i,k}(\mathbf{P}) / P_i \right)$$

where

$$\omega_{i,c} = \begin{cases} 0 & for \quad i \neq c \\ 1 & for \quad i = c \end{cases}$$

END FOR

END FOR

UNTIL $Q_{c,k}(\mathbf{P})$ ARE ALL STABLE (i.e., the changes in $Q_{c,k}(\mathbf{P})$ are small enough)

The Core algorithm employed in the above Linearizer algorithm is as follows:

Algorithm Input:

C: the number of classes of customers

K: the total number of all Type I, II, III service centers

P: the customer population vector

Y: the vector of the sum of the average service time of all Type IV service centers for each customer class

X: the matrix for the average service time for each customer class and for each service center

$\boldsymbol{\lambda}$: the matrix for the $\lambda_{c,k}(\mathbf{P})$ term in Equation (12) for c=1,2,…,C and k=1,2,…,K

Algorithm Output:

$Q_{c,k}(\mathbf{P})$ = the average queue length of class c at center k

The Core algorithm:

// initialization

FOR k = 1 TO K

FOR c = 1 TO C

$Q_{c,k}(\mathbf{P}) = P_c / K$

END FOR

$Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P)$

END FOR

// iterations

REPEAT

FOR c = 1 TO C

FOR k = 1 TO K

$R_{c,k}(\mathbf{P}) = X_{c,k} \cdot [1 + \sum_{i=1}^{C} (P_i - \omega_{i,c}) \cdot (Q_{i,k}(\mathbf{P}) / P_i) + \lambda_{c,k}(\mathbf{P})]$

END FOR

$R_c(\mathbf{P}) = \sum_{k=1}^{K} R_{c,k}(P)$

END FOR

FOR c = 1 TO C

FOR k = 1 TO K

$Q_{c,k}(\mathbf{P}) = \frac{P_c \cdot R_{c,k}(P)}{Y_c + R_c(P)}$

END FOR

$Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P)$

END FOR

UNTIL $Q_{c,k}(\mathbf{P})$ ARE ALL STABLE (i.e., the changes in $Q_{c,k}(\mathbf{P})$ are small enough)

2.3. Performance Prediction for Designing Shared Services

Various AMVA algorithms have been proposed for performance prediction for designed shared services (Rolia *et al.* 2006, Wang 2007, Wang *et al.* 2008, Wang & Wang 2008, Wang *et al.* 2014). Tasks from different operational units or different organizations can be represented as different classes of customers in the queuing network model. Resources in a shared service center can be represented as service centers in the queuing network model. Once an AMVA algorithm solves the queuing network model, we can check whether all specific requirements about the average completion time of different types of tasks are fulfilled. In case that any requirement was not fulfilled, the shared service center should be redesigned by adding more resources. Figure 2 illustrates the flowchart of the aforementioned design process (Wang *et al.* 2014).

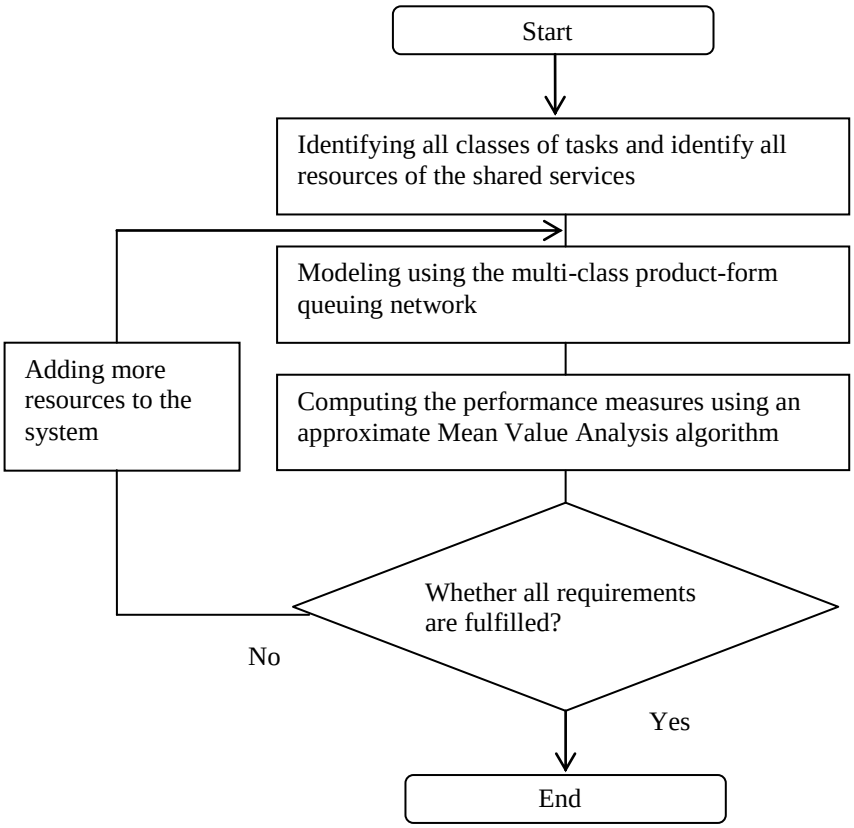

Figure 2. Flowchart for Designing Shared Serviced using AMVA algorithms (Source: (Wang *et al.* 2014))

Chapter 3. A New Family of AMVA Algorithms

In this chapter, we present a new family of AMVA algorithms which we refer to as the Generalized Linearizer algorithms. The Generalized Linearizer algorithms attempt to improve the Linearizer algorithm by achieving better accuracy with the same computational complexities.

The Generalized Linearizer algorithms are based on the approximation

$$Q_k(\mathbf{P}\text{-}\mathbf{1}_c) = \sum_{i=1}^{C} (P_i - \omega_{i,c})^a \cdot (Q_{i,k}(\mathbf{P}) / P_i^b) + \tau_{c,k}(\mathbf{P}) \qquad (16)$$

where both *a* and *b* are real numbers, and

$$\omega_{i,c} = \begin{cases} 0 & for \quad i \neq c \\ 1 & for \quad i = c \end{cases}$$

and the $\tau_{c,k}(\mathbf{P})$ term is an unknown error term. Different values of *a* and *b* result in different Generalized Linearizer algorithms. The Linearizer algorithm is a special case of the Generalized Linearizer algorithms where *a=b=1*.

Theoretically, Equation (16) holds exactly when

$$\tau_{c,k}(\mathbf{P}) = \sum_{i=1}^{C} (P_i - \omega_{i,c})^a \cdot (Q_{i,k}(\mathbf{P}\text{-}\mathbf{1_c}) / (P_i - \omega_{i,c})^a - Q_{i,k}(\mathbf{P}) / P_i^b)$$

Intuitively, Equation (16) is just a rewritten form of Equation (10) of the Linearizer algorithm. While the Linearizer algorithm approximates the $\lambda_{c,k}(\mathbf{P})$ term in Equation (10), the Generalized Linearizer algorithms approximate the $\tau_{c,k}(\mathbf{P})$ term in Equation (16) as

$$\tau_{c,k}(\mathbf{P}) = \tau_{c,k}(\mathbf{P}\text{-}\mathbf{1_j}) \qquad (17)$$

for all j = 1, 2, … C.

Like the Linearizer algorithm, all Generalized Linearizer algorithms employ the Core algorithm as a subroutine which involves solving the following system of nonlinear equations iteratively until convergence:

$$R_{c,k}(\mathbf{P}) = X_{c,k} \cdot [1 + \sum_{i=1}^{C} (P_i - \omega_{i,c})^a \cdot (Q_{i,k}(\mathbf{P}) / P_i^b) + \tau_{c,k}(\mathbf{P})] \qquad (18)$$

$$R_c(\mathbf{P}) = \sum_{k=1}^{K} R_{c,k}(P) \qquad (19)$$

$$Q_{c,k}(\mathbf{P}) = \frac{P_c \cdot R_{c,k}(P)}{Y_c + R_c(P)} \qquad (20)$$

$$Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P) \qquad (21)$$

If the $\tau_{c,k}(\mathbf{P})$ term in Equation (18) is zero and a=b=1, then the Core algorithm is exactly the same as PE algorithm.

Like the Linearizer algorithm, all Generalized Linearizer algorithms iteratively solve the Core algorithm at the original and reduced customer population levels, and estimates and updates the $\tau_{c,k}(\mathbf{P})$ term using Equation (17) for the next round of iteration. As with the Linearizer algorithm, there are two kinds of iterations in a Generalized Linearizer algorithm, inner iterations and outer iterations. The iterations of the Core algorithm are called the inner iterations, and the iterations for estimating and updating the $\tau_{c,k}(\mathbf{P})$ terms are referred to as the outer iterations.

The computational complexities of all Generalized Linearizer algorithms are the same. For all Generalized Linearizer algorithms, the time complexity of the inner iterations is $O(KC^2)$, and the space complexity of the inner iterations is O(KC). Each outer iteration involved O(C) executions of the Core algorithm. Hence, the overall time complexity of a Generalized Linearizer algorithm is $O(KC^3)$ and the space complexity is O(KC).

The pseudo-code of the Generalized Linearizer algorithms is as follows. Note that *a* and

b are not the input parameters of Generalized Linearizer algorithms. Their values are fixed for a given Generalized Linearizer algorithm. Different Generalized Linearizer algorithms employ different values of *a* and *b*.

Algorithm Input:

C: the number of classes of customers

K: the total number of all Type I, II, III service centers

P: the customer population vector

Y: the vector of the sum of the average service time of all Type IV service centers for each customer class

X: the matrix for the average service time for each customer class and for each service center

Algorithm Output:

$R_{c,k}(\mathbf{P})$ = the average response time of a class c customer at center k

$R_c(\mathbf{P})$ = the average response time of a class c customer in the network

$Q_{c,k}(\mathbf{P})$ = the average queue length of class c at center k

$Q_k(\mathbf{P})$ = the average total queue length at center k

The Generalized Linearizer algorithms:

// initialization

FOR k = 1 TO K

FOR c = 1 TO C

$\tau_{c,k}(\mathbf{P}) = 0$

END FOR

END FOR

// iterations

REPEAT

// solving the Core algorithm at the original population level

$\forall c, k : Q_{c,k}(\mathbf{P})$ = Core (C, K, **P**, **Y, X, λ**)

// solving the Core algorithm at the reduced population levels

FOR j=1TO C

$\forall c, k : Q_{c,k}(\mathbf{P\text{-}1_j})$ = Core (C, K, $\mathbf{P\text{-}1_j}$, **Y, X, λ**)

END FOR

// updating the τ matrix

FOR k = 1 TO K

FOR c = 1 TO C

$$\tau_{c,k}(\mathbf{P}) = \sum_{i=1}^{C} (P_i - \omega_{i,c})^a \cdot (Q_{i,k}(\mathbf{P\text{-}1_c}) / (P_i - \omega_{i,c})^a - Q_{i,k}(\mathbf{P}) / P_i^b)$$

where

$$\omega_{i,c} = \begin{cases} 0 & for \quad i \neq c \\ 1 & for \quad i = c \end{cases}$$

END FOR

END FOR

UNTIL $Q_{c,k}(\mathbf{P})$ ARE ALL STABLE (i.e., the changes in $Q_{c,k}(\mathbf{P})$ are small enough)

The Core algorithm involved in the above code is as follows:

Algorithm Input:

C: the number of classes of customers

K: the total number of all Type I, II, III service centers

P: the customer population vector

Y: the vector of the sum of the average service time of all Type IV service centers for each customer class

X: the matrix for the average service time for each customer class and for each service center

$\boldsymbol{\tau}$: the matrix for the $\tau_{c,k}(\mathbf{P})$ term in Equation (12) for c=1,2,…,C and k=1,2,…,K

Algorithm Output:

$Q_{c,k}(\mathbf{P})$ = the average queue length of class c at center k

The Core algorithm:

// initialization

FOR k = 1 TO K

  FOR c = 1 TO C

    $Q_{c,k}(\mathbf{P}) = P_c / K$

  END FOR

  $Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P)$

END FOR

// iterations

REPEAT

FOR c = 1 TO C

FOR k = 1 TO K

$$R_{c,k}(\mathbf{P}) = X_{c,k} \cdot [1 + \sum_{i=1}^{C} (P_i - \omega_{i,c})^a \cdot (Q_{i,k}(\mathbf{P}) / P_i^b) + \tau_{c,k}(\mathbf{P})]$$

END FOR

$$R_c(\mathbf{P}) = \sum_{k=1}^{K} R_{c,k}(P)$$

END FOR

FOR c = 1 TO C

FOR k = 1 TO K

$$Q_{c,k}(\mathbf{P}) = \frac{P_c \cdot R_{c,k}(P)}{Y_c + R_c(P)}$$

END FOR

$$Q_k(\mathbf{P}) = \sum_{c=1}^{C} Q_{c,k}(P)$$

END FOR

UNTIL $Q_{c,k}(\mathbf{P})$ ARE ALL STABLE (i.e., the changes in $Q_{c,k}(\mathbf{P})$ are small enough)

Chapter 4. Experimental Analysis

We experimentally evaluated the actual execution time and accuracy of several Generalized Linearizer algorithms. We compared the new algorithms against the PE and Linearizer algorithms because they are the most popular and most widely used in practice.

Since all Generalized Linearizer algorithms are iterative, their accuracies are affected by the stopping criterion for the iterations. As with the previous studies (Wang & Sevcik 1998, Wang *et al.* 2008), we used the stopping criterion that the maximum change in queue lengths is less than a tolerance, i.e.

$$| Q_{c,k}^{[i+1]}(\mathbf{P}) - Q_{c,k}^{[i]}(\mathbf{P}) | < \varepsilon,$$

where ε is the specified tolerance, and the superscript [i] and [$i+1$] indicates the i^{th} and $i+1^{th}$ iteration respectively.

The error measure for the accuracy of the algorithms that we used in the experiments is the tolerance error defined as

$$e = \max_{c,k} \frac{| Q_{c,k}(\mathbf{P}) - Q^*_{c,k}(\mathbf{P}) |}{P_c},$$

where $Q^*_{c,k}(\mathbf{P})$ is the exact queue length of class c customers at server k computed by the MVA algorithm, and $Q_{c,k}(\mathbf{P})$ is the approximate value computed by a Generalized Linearizer algorithm.

4.1.Experiment Settings

In our experiments, five hundred multi-class product-form queuing network models were randomly generated for each C=1, 2, 3, and 4. The parameters for generating these models are shown in Table 1. These randomly generated models were solved by the MVA algorithm and the Generalized Linearizer algorithms.

We could not choose larger values for K and C in our experiments which require solving the queuing network models using the MVA algorithm. It is not practical to solve larger queuing network models using the MVA algorithm as it could take more than a month to compute the solution.

Parameters	Distributions / Values	
Number of queuing network models	500	
K	Uniform (2,30)	
X (all entries in this matrix)	Uniform (0.1, 20.0)	
Y (all entries in this vector)	Uniform (0.0, 100.0)	
C	1, 2, 3, 4	
P (all entries in this vector)	C=1	P_1: Uniform (3,30)
	C=2	P_1: Uniform (3,30) P_2: Uniform (3,30)
	C=3	P_1: Uniform (3,30) P_2: Uniform (3,30) P_3: Uniform (3,30)
	C=4	P_1: Uniform (3,30) P_2: Uniform (3,30) P_3: Uniform (3,30) P_4: Uniform (3,30)
ε (tolerance for iterations)	0.0001	

Table 1. Parameters for generating multi-class product-form queuing network models

4.2.Experiment Results on Accuracy of the Algorithms

4.2.1. Different values of *a* and *b*

The Generalized Linearizer algorithms are a family of algorithms. Different values of *a* and *b* result in different instances of the algorithms. Clearly, the choice of the values of *a* and *b* affects the accuracy of the Generalized Linearizer algorithm. As shown in Figure 3, we found that the Generalized Linearizer algorithms with *b* close to *a* are more

accurate than those with *b* much larger than *a*. Figure 3 shows the mean tolerance errors of different Generalized Linearizer algorithms with a=1 and different values of *b*. The reason is because when the difference between *a* and *b* is small, the absolute value of the $\tau_{c,k}(\mathbf{P})$ term in Equation (16) is small. As the result, the difference between the exact value of the $\tau_{c,k}(\mathbf{P})$ term and the approximated value also tends to be small, which translated into a small error. Hence, the Generalized Linearizer algorithms with *a=b* are likely to be more accurate than those with *a≠b*.

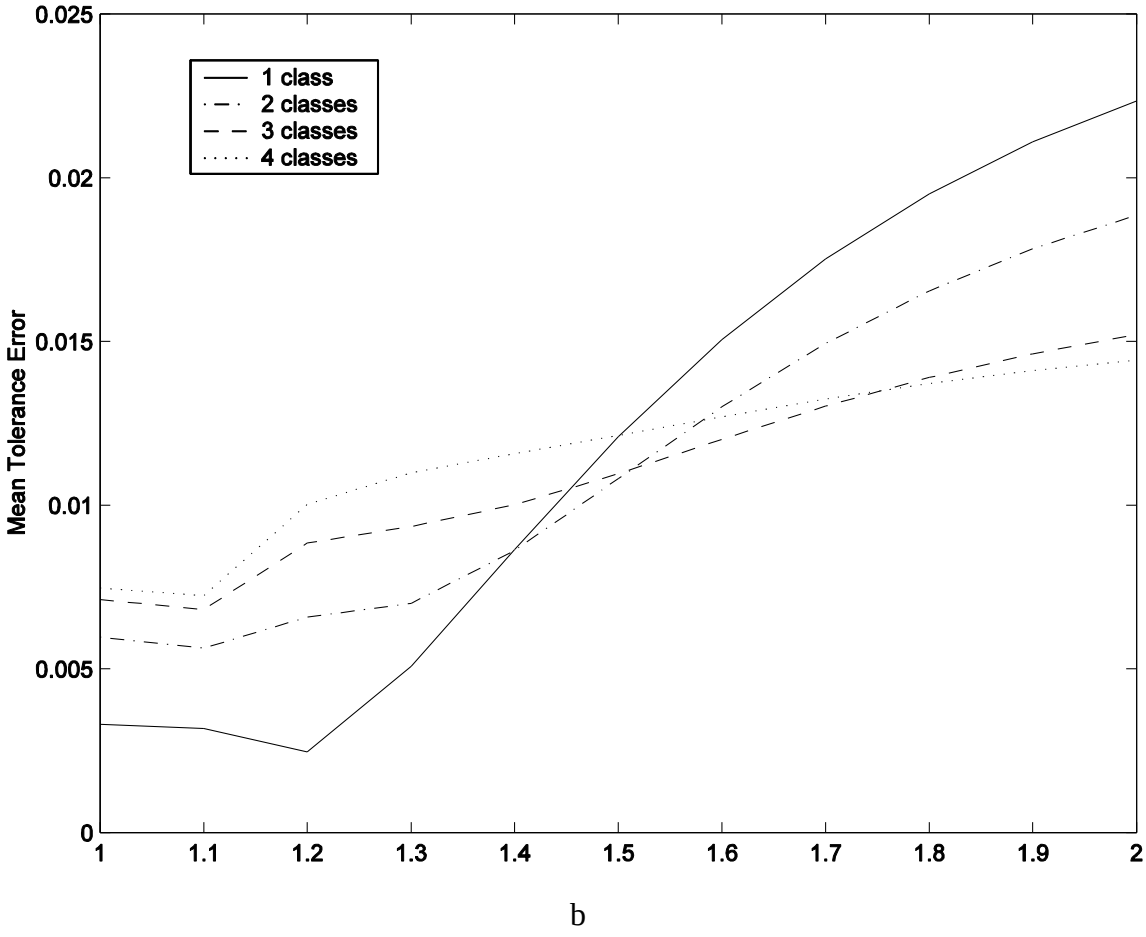

Figure 3. The mean tolerance errors for the Generalized Linearizer algorithms with *a=1* and different values of *b*.

4.2.2. Different Generalized Linearizer algorithms

Based on the experiment results of the previous subsection, the Generalized Linearizer algorithms with $a=b$ are likely to be more accurate than those with $a\neq b$. As the Linearizer algorithm is a Generalized Linearizer algorithm with $a=b=1$, we investigated the accuracy of various Generalized Linearizer algorithms in order to find a more accurate algorithm than the Linearizer algorithm. Table 2 shows the mean tolerance errors and max tolerance errors for various algorithms. As shown in Table 2, we found that the Generalized Linear algorithms with $a=b=1.5$, $a=b=2.0$, $a=b=3.0$, $a=b=4.0$ are more accurate than the Linearizer algorithm and PE algorithm. We also found that as the number of customer classes increases, the superiority of these Generalized Linearizer algorithms over the Linearizer algorithm in terms of accuracy increases. Our results are consistent with the previous results (Wang *et al.* 2008).

Algorithms	Tolerance Errors	Models			
		C=1	C=2	C=3	C=4
PE	***Mean(e)***	0.011	0.019	0.023	0.031
	Max(e)	0.054	0.056	0.059	0.062
Linearizer (*a=b=1*)	***Mean(e)***	0.003	0.006	0.007	0.008
	Max(e)	0.019	0.023	0.026	0.033
Generalized Linearizer (*a=b=1.5*)	***Mean(e)***	0.002	0.004	0.004	0.004
	Max(e)	0.014	0.014	0.014	0.017
Generalized Linearizer (*a=b=2.0*)	***Mean(e)***	0.002	0.002	0.003	0.003
	Max(e)	0.008	0.010	0.010	0.011
Generalized Linearizer (*a=b=3.0*)	***Mean(e)***	0.001	0.002	0.003	0.004
	Max(e)	0.011	0.017	0.015	0.015
Generalized Linearizer (*a=b=4.0*)	***Mean(e)***	0.001	0.004	0.006	0.007
	Max(e)	0.021	0.017	0.020	0.048

Table 2. Mean tolerance errors and maximum tolerance errors for different Generalized Linearizer algorithms and the PE algorithm

4.3.Experiment Results on Computation Time of the Algorithms

One important advantage of the AMVA algorithms is that they can compute the solutions quickly. We measured the total execution time of each algorithm in Table 2 for the 500 multi-class product-form queuing network models in Table 1. The experiments were conducted on a PC with 4GB memory running Windows XP. Table 3 shows the total execution time of the algorithms in terms of the CPU seconds. As shown in Table 3, we found that the Generalized Linearizer algorithms achieve similar execution time to the

Linearizer algorithm. We expect that as the number of service centers and the number of classes increase, the execution time of the MVA algorithm will increase exponentially and the AMVA algorithms will be much faster than the MVA algorithm. The PE algorithm is faster than the Generalized Linearizer algorithms as expected as it is much less accurate.

Algorithms	**Models**			
	C=1	**C=2**	**C=3**	**C=4**
MVA	0.10	2.26	30.32	650.86
PE	0.09	1.34	3.48	5.61
Linearizer (*a=b=1*)	2.12	5.13	28.37	420.15
Generalized Linearizer (*a=b=1.5*)	2.02	4.83	21.69	354.97
Generalized Linearizer (*a=b=2.0*)	2.08	3.56	20.18	331.01
Generalized Linearizer (*a=b=3.0*)	3.81	5.05	19.84	329.79
Generalized Linearizer (*a=b=4.0*)	4.01	4.98	28.12	301.28

Table 3. The execution time of the algorithms for the 500 randomly generated models

Chapter 5 Conclusions and Future Work

In this thesis, we have discussed two types of shared services, internal and external shared services. We have examined the important issues with the design and implementation of shared services. One of the issues is to ensure all requirements of the completion time of different types of tasks are fulfilled. We have proposed to use multi-class product-form queuing network models to predict the performance of shared service centers during the design of shared services. As various AMVA algorithms had been proposed to predict the completion time of different types of tasks in a shared service center, we have proposed the Generalized Linearizer algorithms. The Generalized Linearizer algorithms are a family of AMVA algorithms. The previously proposed Linearizer algorithm is just a special instance of this family. The Generalized Linearizer algorithms are capable of effectively predicting the performance of shared service centers, and ensuring all requirements of the completion time of different types of tasks are fulfilled during the design of shared services. We have also examined the computational time of the Generalized Linearizer algorithms as well as their accuracies. Based on our experimental results, we have suggested various algorithms in the family of the

Generalized Linearizer algorithms that can achieve better accuracy than the Linearizer algorithm with comparable computational time.

In the future, we plan to extend our study and conduct further experiments to explore the properties of the Generalized Linearizer algorithms. We plan to identify the best algorithm in the family of the Generalized Linearizer algorithms in terms of computational time and accuracy.

References

Ask, A., Hatakka, M., & Gronlund, A. (2008). The Orebro city citizen-oriented e-government strategy. *International Journal of Electronic Government Research*, *4*(4), 69-88.

Balbo, G., & Serazzi, G. (1996). Asymptotic analysis of multiclass closed queueing networks: Common bottleneck, *Performance Evaluation*, 26 (1), 51-72.

Balbo, G., & Serazzi, G. (1997). Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks, *Performance Evaluation*, 30 (3), 115-152.

Bard, Y. (1979). Some extensions to multiclass queueing network analysis, in: M. Arato, A. Butrimenko, E. Gelenbe (Eds.), *Performance of Computer Systems*, pp.51-62, North-Holland, Amesterdam, Netherlands.

Baskett, F., Chandy, K. M., Muntz, R. R., & Palacios, F. G. (1975). Open, closed, and mixed networks of queues with different classes of customers, *Journal of the ACM*, 22(2), 248-260.

Bolch, G., Greiner, S., Meer, H., & Trivedi, K. S. (2006). *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*,

2nd edition, Wiley-Interscience, Hoboken, New Jersey.

Borman, M. (2010). Characteristics of a successful shared services centre in the Australian public sector. *Transforming Government: People, Process and Policy*, *4*(3), 220-231.

Buzen, J. P. (1973). Computational algorithms for closed queueing networks with exponential servers, *Communications of the ACM*, 16 (9), 527-531.

Chow, W. M. (1983). Approximations for large scale closed queueing networks, *Performance Evaluation*, 3(1), 1-12.

Chandy, K. M. & Neuse, D. (1980). Linearizer: A heuristic algorithm for queueing network models of computing systems, *Communications of the ACM*, 23(10), 573-583.

Chandy, K. M., & Sauer, C. H. (1986). Computational algorithm for the exact analysis of multiple-chain closed queueing networks, *Journal of the ACM*, 33(4), 768-791.

Coway, A. E. & Georganas, N. D. (1986). RECAL -- a new efficient algorithm for the exact analysis of multiple-chain closed queueing networks, *Journal of the ACM*, 33(4), 768-791.

Conway, A. E. & De Souza e Silva, E., & Lavenberg, S. S. (1989). Mean value analysis by chain product form queueing networks, *IEEE Transactions on Computers*, C-38(3), 432-442.

Cremonesi, P., Schweitzer, P. J., & Serazzi, G. (2002). A unifying framework for the approximate solution of closed multiclass queueing networks, *IEEE Transactions on Computers*, C-51(12), 1423-1434.

Davenport, T. H., Harris, J. G., & Cantrell, S. (2004). Enterprise systems and ongoing process change. *Business Process Management Journal*, *10*(1), 16-26.

Davenport, T. H., & Short, J. (1990). The new industrial engineering. *Sloan Management Review, 31*(4), 11-27.

Denning, P. J., & Buzen, J. P. (1978). The operational analysis of queueing network models, *Computing Surveys,* 10 (3), 225-261.

Eager, D. L. & Sevcik, K. C. (1984). Analysis of an approximation algorithm for queueing networks, *Performance Evaluation*, 4(4), 275-284.

Eager, D. L. & Sevcik, K. C. (1986). Bound hierarchies for multiple-class queueing networks, *Journal of the ACM*, 33(1), 179-206.

Gulati, R., & Singh, H. (1998). The architecture of cooperation: managing coordination costs and appropriation concerns in strategic alliance. *Administrative Science Quarterly*, *43*(4), 781-814.

Hoyme, K. P., Bruell, S. C., Afshari, P. V., & Kain, R. Y. (1986). A tree-structured mean value analysis algorithm, *ACM Transactions on Computer Systems*, 4(2), 178-185.

Hsieh, C. T., & Lam, S. S. (1988). PAM -- a noniterative approximate solution method for closed multichain queueing networks, *ACM SIGMETRICS Performance Evaluation Review,* 16(1), 261-269.

Janssen, M., & Joha, A. (2006). Motives for establishing shared service centers in public administrations. *International Journal of Information Management*, *26*(2), 102-112.

Kakabadse, A., & Kakabadse, N. (2000). Sourcing: New face to economies of scale and the emergence of new organizational forms. *Knowledge and Process Management*, *7*(2), 107-118.

Kamal, M. M. (2012). Shared services: lessons from private sector for public sector domain. *Journal of Enterprise Information Management*, *25*(5), 431-440.

King, D. R. (2006). Implications of uncertainty on firm outsourcing decisions. *Human*

Systems Management, *25*(2), 115-124.

Lam, S. S. (1983). A simple derivation of the MVA and LBANC algorithms from the convolution algorithm, *IEEE Transactions on Computers*, C-32(11), 1062-1064.

Lam, S. S., & Lien, Y. L. (1983). A tree convolution algorithm for the solution of queueing networks, *Communications of the ACM*, 26(3), 203-215.

Lavenberg, S. S. & Reiser, M. (1980). Stationary state probabilities of arrival instants for closed queueing networks with multiple types of customers, *Journal of Applied Probability,* 17(4), 1048-1061.

Lindvall, J., & Iveroth, E. (2011). Creating a global network of shared service centres for accounting. *Journal of Accounting & Organizational Change*, *7*(3), 278-305.

McDowell, J. (2011). Shared services centers can drive significant savings. *Healthcare Financial Management*, *65*(6), 118-124.

Minnaar, R. A. (2013). Shared service centres and management control structure change. *Journal of Accounting & Organizational Change*, *9*(1), 74-98.

Niehaves, B., & Krause, A. (2010). Shared service strategies in local government - a multiple case study exploration. *Transforming Government: People, Process and*

Policy, *4*(3), 266-279.

Pattipati, K. R., Kostreva, M. M. & Teele, J. L. (1990). Approximate mean value analysis algorithms for queueing networks: Existence, uniqueness, and convergence results, *Journal of the ACM*, 37(3), 643-673.

Reiser, M. & Kobayashi, H. (1975). Queueing networks with multiple closed chains: Theory and computational algorithms, *IBM journal of Research and Development*, 19(3), 283-294.

Reiser, M. & Lavenberg, S. S. (1980). Mean value analysis of closed multichain queueing networks, *Journal of the ACM*, 27(2), 313-322.

Rolia, J., Cherkasova, L., Arlitt, M., & Machiraju, V. (2006). Supporting application quality of service in shared resource pools. *Communications of the ACM*, *49*(3), 55-60.

Ulbrich, D. (1995). Shared services: From vogue to value. *Human Resource Planning*, *18*(3), 12-23.

Ulbrich, F. (2006). Improving shared service implementation: Adopting lessons from the BPR movement. *Business Process Management Journal*, *12*(2), 191-205.

Ulbrich, F. (2010). Adopting shared services in a public-sector organization. *Transforming Government: People, Process and Policy*, 4(3), 249-265.

Wang, H. (2007). Performance Analysis for Shared Services. *Communications of the International Information Management Association*, 7(2), 61-68.

Wang, H.**,** & Sevcik, K. C. (2000). Experiments with improved approximate mean value analysis algorithms. *Performance Evaluation*, *39*(1-4), 189-206.

Wang, H., & Wang, S. (2008). An Approximate Queuing Network Analysis Method for Capacity Planning of Shared Services. *Proceedings of the 29th Annual Conference of the Administrative Sciences Association of Canada, Management Science Division*, pp.6-15. Halifax, NS: Administrative Sciences Association of Canada.

Wang, H., & Wang, S. (2014). Ontological Map of Service Oriented Architecture for Shared Services Management. *Expert Systems with Applications*, 41(5): 2362-2371.

Wang, H., Liu. Y., Jiang, Y., and Wang, S. (2014). Queuing Networks for Designing Shared Services. In: Wang, J. (Ed.), Encyclopedia of Business Analytics and Optimization, pp. 256-261. Hershey, PA: IGI Global.

Wang, S., & Wang, H. (2007). Shared services beyond sourcing the back offices:

Organizational design. *Human Systems Management*, *26*(4), 281-290.