ON FORK-JOIN QUEUES AND MAXIMUM RATIO CLIQUES

A Dissertation

by

SAMYUKTA SETHURAMAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Natarajan Gautam
Co-Chair of Committee,	Sergiy Butenko
Committee Members,	Kiavash Kianfar
	Sivakumar Rathinam
Head of Department,	César Malavé

December 2015

Major Subject: Industrial Engineering

ABSTRACT

This dissertation consists of two parts. The first part delves into the problem of response time estimation in fork-join queueing networks. These systems have been seen in literature for more than thirty years. The estimation of the mean response time in these systems has been found to be notoriously hard for most forms of these queueing systems. In this work, simple expressions for the mean response time are proposed as conjectures. Extensive experiments demonstrate the remarkable accuracy of these conjectures. Algorithms for the estimation of response time using these conjectures are proposed. For many of the networks studied in this dissertation, no approximations are known in literature for estimation of their response time. Therefore, the contribution of this dissertation in this direction marks significant progress in the analysis of fork-join queues.

The second part of this dissertation introduces a fractional version of the classical maximum weight clique problem, the maximum ratio clique problem, which is to find a maximal clique that has the largest ratio of benefit and cost weights associated with the cliques vertices. This problem is formulated to model networks in which the vertices have a benefit as well as a cost associated with them. The maximum ratio clique problem finds applications in a wide range of areas including social networks, stock market graphs and wind farm location. NP-completeness of the decision version of the problem is established, and three solution methods are proposed. The results of numerical experiments with standard graph instances, as well as with real-life instances arising in finance and energy systems, are reported.

DEDICATION

To Amma and Appa

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to my advisor, Dr. Natarajan Gautam. He has been extremely patient and supportive throughout the course of my PhD. His genuine happiness in my achievements, and encouragement when my spirits were low kept me going through these years. I would also like to thank my co-advisor, Dr. Sergiy Butenko. The energy and high spirits with which he approaches any task at hand is something that I wish to incorporate in my life as well. I consider myself very fortunate to have two advisors who have always put my interests in the forefront while making any decision. I would also like to thank my committee members, Dr. Kiavash Kianfar and Dr. Sivakumar Rathinam, for their valuable inputs to my research and dissertation writing.

I would like to express my gratitude towards my undergraduate advisor, Dr. Prathap Haridoss. It was his encouragement and belief in my abilities that motivated me to apply for a PhD program in an area that was different from my undergraduate major. Sincere thanks are also due to Dr. G. Srinivasan who introduced the field of operations research to me. The courses taught by him planted the first seeds of interest in this field that continues to fascinate me up till now.

The faculty and staff of my department have always been there to help me whenever I needed anything. I thoroughly enjoyed the courses taught by many faculty members and I am grateful for having had the opportunity to attend them. I would like to thank Ms. Judy Meeks and Ms. Erin Roady for patiently clarifying and working with me on each and every procedure required for the fulfillment of my degree. Ms. Judy Meeks was also a trusted confidant to me. I could comfortably approach her with matters that I was unsure whom to talk to about. I would also like to

thank the department for providing me with a teaching assistantship and the Energy Institute for the MP2 Energy Fellowship.

I have made many valuable friends during my PhD life and I am grateful to each and every one of them for making this time so enjoyable for me. I would like to thank Kaarthik Sundar in particular, for patiently helping me with LaTeX and C++ when I was getting started on them. The TAMU Student Chapter of INFORMS provided the means to meet many of these friends and I am happy to have been a member of this organization. I am also grateful to the amazing friends that I made by being a member of the Rangeela dance group. The numerous evenings that we spent in practice sessions count among some of my happiest moments in College Station and will remain in my memory for a very long time to come.

Finally, I would like to thank my parents and sisters. My parents have made numerous sacrifices for me and my education and I can always count on my sisters to cheer me on in any task that I take up. This dissertation would not have been possible without their support.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGEMENTS iv

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES ix

1. INTRODUCTION 1

1.1 Overview 1
1.2 Response Time in Fork-Join Queues 2
1.2.1 Applications 4
1.3 The Maximum Ratio Clique Problem 8
1.4 Organization 9

2. SYMMETRIC n-DIMENSIONAL FORK-JOIN QUEUES 10

2.1 Literature Review 10
2.2 Problem Description 12
2.3 Response Time Estimation 15
2.3.1 Exponential Service Times 18
2.4 Response Time Computation Algorithm for Non-Exponential Service Times 20
2.4.1 Numerical Example 21
2.5 Results and Comparisons 23
2.5.1 Comparison with Simulations 33
2.5.2 Approximation by Nelson and Tantawi 34
2.5.3 Approximation by Ko and Serfozo 37
2.5.4 Approximation by Thomasian and Tantawi 38
2.6 Conclusions 39

3. MORE ON FORK-JOIN QUEUEING NETWORKS 41

3.1 Literature Review . . . 41
3.2 Symmetric Tandem Fork-Join Queueing Network . . . 43
3.2.1 Response Time Estimation . . . 45
3.2.2 Numerical Example . . . 49
3.2.3 Experimental Results . . . 49
3.3 Heterogeneous Fork-Join Queueing Systems . . . 53
3.3.1 Problem Description . . . 53
3.3.2 Response Time Estimation . . . 53
3.3.3 Numerical Example and Results . . . 55
3.4 (n,k) Fork-Join Queues . . . 57
3.4.1 Comparison with Simulations . . . 60
3.4.2 Conclusions . . . 61

4. The Maximum Ratio Clique Problem . . . 63

4.1 Introduction . . . 63
4.2 Computational Complexity . . . 67
4.3 Integer Programming Formulation . . . 71
4.4 Solution Methods . . . 72
4.4.1 Linearization . . . 72
4.4.2 Binary Search . . . 73
4.4.3 Newton's Method . . . 76
4.5 Results of Computational Experiments . . . 77
4.5.1 Description of Test Instances . . . 78
4.5.2 Comparison of Results . . . 80
4.6 Conclusion . . . 94

5. CONCLUSIONS AND FUTURE WORK . . . 95

REFERENCES . . . 99

LIST OF FIGURES

FIGURE		Page
1	Fork-Join Queueing Network	3
2	Symmetric n-Dimensional Fork-Join Queueing Network	13
3	Plot of simulated average response time vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $\mu = 1$ and exponential service time distribution	18
4	Plot of parameter m_n vs. $\log(n)$	19
5	Plot of simulated and predicted average response times vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $\mu = 1$ and exponential service time distribution	21
6	Plot of simulated and predicted average response times vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $\mu = 1$ and Erlang-2 service time distribution	22
7	Symmetric Tandem Fork-Join Queueing Network	44
8	Plot of simulated average response time vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $l = 5$ and $\mu = 1$	48
9	Plot of simulated and predicted average response times vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $l = 5$ and $\mu = 1$	49
10	Plot of $E[T^{(1)}_{(5,3)}]$ vs. $E[T_{(5,3)}]$	60

LIST OF TABLES

TABLE		Page
1	List of Notations	15
2	Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 3$ and exponential service time distribution	24
3	Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 5$ and exponential service time distribution	24
4	Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 10$ and exponential service time distribution	25
5	Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 20$ and exponential service time distribution	25
6	Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 30$ and exponential service time distribution	26
7	Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 40$ and exponential service time distribution	26
8	Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 50$ and exponential service time distribution	27
9	Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 3$ and Erlang-2 service time distribution	27
10	Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 5$ and Erlang-2 service time distribution	28

11 Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 10$ and Erlang-2 service time distribution 28

12 Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 15$ and Erlang-2 service time distribution 29

13 Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 20$ and Erlang-2 service time distribution 29

14 Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 30$ and Erlang-2 service time distribution 30

15 Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 2$ and Pareto service time distribution 30

16 Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 3$ and Pareto service time distribution 31

17 Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 5$ and Pareto service time distribution 31

18 Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 7$ and Pareto service time distribution 31

19 Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 10$ and Pareto service time distribution 32

21 Comparison with simulation results for $n = 2$, $l = 2$ and $n = 2$, $l = 3$ 51

22 Comparison with simulation results for $n = 2$, $l = 4$ and $n = 2$, $l = 5$ 51

23 Comparison with simulation results for $n = 5$, $l = 2$ and $n = 5$, $l = 3$ 51

24 Comparison with simulation results for $n = 5$, $l = 4$ and $n = 5$, $l = 5$ 52

25 Comparison with simulation results for $n = 10$, $l = 2$ and $n = 10$, $l = 3$ 52

26 Comparison with simulation results for $n = 10$, $l = 4$ and $n = 10$, $l = 5$ 52

27 Comparison with simulation results for $n = 2$, $\mu_1 = 1.0$ and $\mu_2 = 1.5$. 58

28 Comparison with simulation results for $n = 3$, $\mu_1 = 0.5$, $\mu_2 = 1.0$ and $\mu_3 = 1.5$ 58

29 Description of test instances used, optimal objective function values, and the corresponding maximal clique sizes. 83

30 Comparison of the results of experiments using the proposed approaches on the graph instances described in Table 29. 88

1. INTRODUCTION

1.1 Overview

Mankind is constantly on the look out for methods to achieve more benefit while utilizing lesser resources. Accomplishment of this ideology however, can happen in a very wide variety of ways and forms the basis of the entire discipline of operations research. Many interesting research problems have been studied to this end since the late nineteenth century. Now, in the twenty-first century, due to the rapid pace of advent of new technology, there is more scope for research on improvement in system efficiency than ever before with new problems arising every day. Solutions to these problems provide guidelines for making decisions on input parameters that are within human control and which will result in an efficient system.

To be able to construct a processing system which generates the maximum possible benefit, it is important to know how much benefit is gained given the resource availability in the form of input parameters and how this benefit changes with change in the parameters. However, this basic information can prove to be extremely hard to obtain. The processing systems analyzed in the first part of this dissertation fall into this category. In many systems obtaining this information is enough to determine the input parameters that will result in the most efficient system possible. However, there exist systems in which the relationship between the benefit gained and the input parameters is a function that turns out to be so complex that it becomes extremely hard to see which input parameters are the best for a given scenario. The system analyzed in the later part of this dissertation falls into the second category. Our contribution lies in giving a hitherto unavailable set of guidelines in the form of algorithms to enable the controlling authority to make informed decisions. We

now describe these two kinds of systems in detail. The next section serves as an introduction to Chapters 2 and 3.

1.2 Response Time in Fork-Join Queues

Time is one of the most precious resources available to mankind. Parallel processing of tasks is a method that humans have come up with to make efficient utilization of time. To gain the most out of parallel processing systems, it is important to know how their performance in general and time reduction in particular would change with change in parameters of the system. However, this relationship between input parameters and processing time is not necessarily straightforward. This is true especially if there are waiting times and queues being formed in the system.

Day-to-day situations faced by people have been successfully modeled as queueing systems since the early 20th century [22]. In the rich queueing literature that has developed since then, there are a very wide range of problems that have been analyzed. Some systems like the $M/M/s$ queueing system have been shown to be simple enough to have a closed form solution. However, even for systems that look and sound relatively simple, like the $G/G/1$ queue, it has not been possible to derive closed form solutions yet. For such systems, approximations under general and/or heavy traffic conditions, bounds and other properties have been studied by researchers over the years.

The phenomenon of queueing is observed in many parallel processing systems. An arriving job to such a system is partitioned into tasks being processed in parallel, some or all (depending on the system definition) of which need to be completed to finish processing the job. Jobs arrive into this system at random time intervals. If a particular task belonging to an arriving job finds that task of another job being processed, it will wait in queue for its turn to be processed. Each parallel processor

processes tasks one by one and does not coordinate with other processors. Therefore, it is possible that one task of a job is finished but another task is not, even though all the processors operate on a first come first served service discipline. When this happens, the job waits for the last of its tasks to get done, to exit the system. This queueing system is called the fork-join queueing system. Fig. 1 shows a fork-join queueing network with n tasks. Arriving jobs are partitioned instantaneously at the fork (F) station and each task joins its own queue. Once the processing of a task is complete, it waits in a buffer for other tasks belonging to the same job to be completed before joining to complete the job at the join (J) station. The typical quantity of interest in this scenario is the average time spent in the system by a job, known as the mean response time of the system. This quantity is known to be notoriously hard to find. In this research we provide some new insights that will help us to derive a numerical approach to estimate the mean response time of the system.

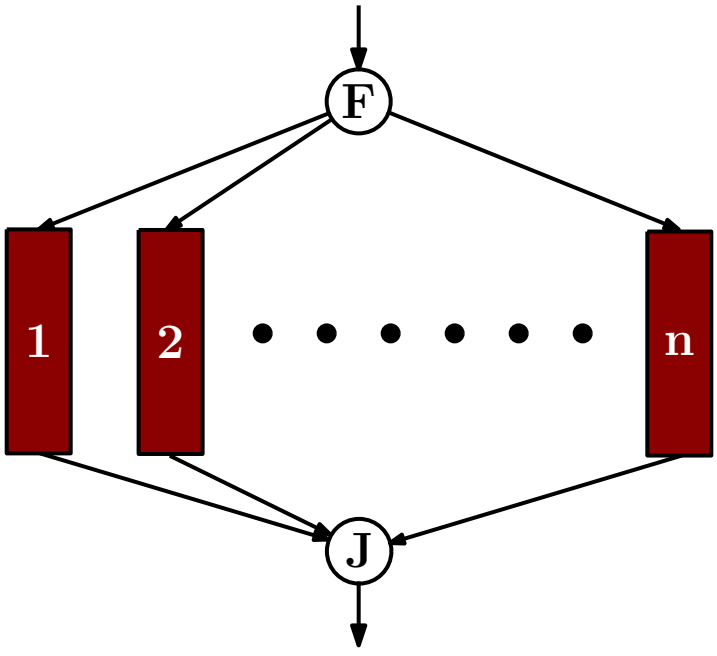

Figure 1: Fork-Join Queueing Network

1.2.1 Applications

Fork-join queueing networks have a very diverse set of applications ranging from manufacturing systems to health care and wireless sensor networks. We enumerate some of these here.

1. **Network Coding:** The interest in fork-join queueing systems first arose when parallel processing computer systems became popular [38]. Data in computer systems is stored on multiple disks (disk arrays) and is accessed in parallel to improve efficiency. The same data might be replicated onto multiple disks to improve the reliability of the system [46], [13], [45], [28]. When a request for a file is made to this system, this corresponds to arrivals to the queueing system. The request requires data from multiple disks in the array and each disk would process requests in the order in which they arrive. The response time of the system is the time needed to satisfy the request made. This parallel accessing of data from multiple disks leads to a fork-join queue. This scenario is also seen in RAID5 disk arrays when one of the disks fail. In the coding of disk arrays, fork-join queues are also encountered at times when one of the disks fail in which case, the data on the disk is also encrypted in a distributed fashion among other disks in the array and response from all the disks is needed to complete the request [56]. This distributed storage is made more effective using network coding which again leads to fork-join queueing systems in a similar manner [32].

2. **Manufacturing Systems:** One of the most well-known examples of a fork-join queueing network is that of parallel assembly lines in manufacturing systems. We present a simple example from [49]. Consider a shop that manufactures coats. An arrival of a job into the system would correspond to an order

for a coat given by a customer. The job of stitching this coat might consist of tasks that can be processed in parallel. For example, if the shop is adequately staffed, the sleeves, fronts, backs, collars, pockets, lining etc. could be prepared at the same time. Once all the above are ready the coat is ready for the final assembly which corresponds to the join process. After the final assembly, the coat can be delivered to the customer which corresponds to the job leaving the system. If there are multiple orders pending, the other tasks of other orders would wait in queue until that task of all the previous orders are processed. This is a simple example of a fork-join queueing system in a manufacturing facility. More complex systems involving multiple sub-tasks within one task might be necessary in more sophisticated processes like car assembly. If the product requires the same server to process multiple tasks, this results in a multi-class fork-join queueing system [29].

3. **Health Care Systems:** In the area of healthcare, applications of fork join queues exist in the process of a physician arriving at a diagnosis. A patient entering a hospital for consultation corresponds to arrivals to the system. The physician might need to prescribe a series of tests which could be performed in parallel. The processing of the test sample would probably be conducted on a first come first served basis. Each of the tests corresponds to a parallel task. Once all the test results are available the physician arrives at a diagnosis which corresponds to the job exiting the system.

   In a different setting, mental processes have been modeled as fork-join queueing networks by Liu [42]. Stimulants that require a response from the human body are modeled as customers to the queueing system and the author gives instances where the brain processes different issues associated with

the stimulant in parallel. The response time is the time between when the stimulant is encountered and when the human body responds to it.

4. **Global Data Collection:** Existing theory on fork-join queues has been used in global data collection and analysis by Wu et al [63]. This type of data analysis is important in the context of global supply chains. In this case, the online algorithm requests data from a data generating center such as a raw material supplier and transfers that to a center that combines this data with data from other centers to perform data analysis. The result of this analysis is then sent to the next processing center. The request for data from a data generating center corresponds to arrivals to the system. The collection of the requested data results in a finite service time and any pending requests wait in a virtual queue for their turn to be processed. Wu et al [63] use existing approximations for fork-join queues in the literature to estimate the response time of this system and make appropriate decisions on the design of the algorithm for data analysis.

5. **Wireless Sensor Networks:** Consider a wireless sensor network with spatially distributed sensors. A real life example where fork join queues are observed in wireless sensor networks is in cities such as Singapore. Sensors capable of sensing environmental factors such as temperature, humidity, air pollution and noise level are installed on lamp posts in the city. The controlling authority is interested in only a function of the data being collected, for example the maximum temperature. All the sensors sense at exactly the same instants of time. The time interval between consecutive sensing instants of time is random with a known distribution. This sensing of data corresponds to arrivals to the system. Once a set of data has been sensed, each sensor needs to transmit

it to the controlling authority. The sensors can transmit only one data point at a time. It takes a random amount of time with a known distribution to complete the transmission of one data point from a sensor to the end user. This corresponds to the service time of the system. If a set of data are sensed while a transmission is taking place at a sensor, the new data point waits in queue in a buffer at the sensor for its turn to be transmitted. For each set of data that has been sensed at the same time, the controlling authority waits to receive data points from each sensor before computing the function of the data. Computing the function takes a negligible amount of time. Once the function of a set of data that has been sensed simultaneously has been computed, the information is stored and this corresponds to departure of the entity from the system.

Fork join queues arise in wireless sensor networks in many other situations [7]. Wireless sensors can be used to monitor water and electricity utilization in smart cities. Similar to the previous case, fork-join queues are encountered in the analysis of this utilization. In environmental applications, wireless sensors are installed at places that are hard to reach and can be used for monitoring of wildlife habitats and forest fires. For monitoring of forest fires for example, knowing the maximum temperature among temperatures measured by a set of sensors in an area might be sufficient. Wireless sensors are also employed in security applications for surveillance at airports and borders, where again fork-join queues are encountered.

The analysis of the time required by fork-join queueing systems to complete processing a task forms the first part of this dissertation. The second part considers a problem in which the benefit attained is easily expressed as a function of the input

parameters. However, the function is so complex that it becomes extremely hard to find the set of parameters that will result in the most efficient performance. In the next section we give an overview of this system. A detailed introduction and analysis is presented in Chapter 4.

1.3 The Maximum Ratio Clique Problem

Natural assets and money are another set of resources apart from time, which become highly important due to their limited availability in most situations. Any productive system requires the investment of these resources in some form to obtain benefit in some other form. The ratio of the benefit to the investment determines the overall value gained per unit resource and is an important measure of the performance of the system. To enable the controlling authority to make informed decisions, this ratio is expressed in terms of the input parameters to the system. However, once this expression is obtained, if the number of input parameters is high, it can become very difficult to determine the combination of parameters that will result in the highest benefit to cost ratio.

Combinatorial optimization is an area of study dealing with choosing the best possible set of parameters subject to their feasibility. Many classical problems in this field have been studied over the years. However, use of combinatorial optimization to maximize a function that is in the form of a ratio of functions of input parameters is a relatively new field of study. Therefore, it is rich with prospects of formulating and solving new problems. We formulate one such problem, known as the *Maximum Ratio Clique Problem*.

The input parameters in the expression for the benefit to cost ratio represent the utilization of available resources. Often, these resources are such that the use of one prohibits the use of another. The solution to the maximum ratio clique problem

finds a set of compatible resources that maximizes the benefit to cost ratio of the system under consideration. Similar to fork-join queues, this problem also has a very wide range of applications including but not limited to social network analysis, stock markets and establishment of wind farms.

In the second part of this dissertation, we formulate and conduct a thorough analysis of the maximum ratio clique problem.

1.4 Organization

The rest of this dissertation is organized as follows: In Chapter 2, we propose a new and highly efficient technique to estimate the response time in the basic form of fork-join queues. In Chapter 3, we extend this technique to more complex forms of fork-join queues for which there are no response time estimation methods available in literature to the best of our knowledge. Chapter 4 deals with formulation and solution methods of the maximum ratio clique problem. Finally, we conclude this dissertation in Chapter 5.

2. SYMMETRIC n-DIMENSIONAL FORK-JOIN QUEUES

In this chapter we consider the basic and most well-studied form of fork-join queues, the symmetric n-dimensional fork-join queueing system. Arriving jobs are partitioned into n stochastically identical tasks each of which need to be completed before the job can exit the system. The arrivals are according to a Poisson process and service times at each task are independent, identical and follow a known distribution. We present a highly efficient approximation technique to estimate the mean response time of this system. We now provide a review of the advances made in literature with respect to understanding these queueing systems.

2.1 Literature Review

Work on fork-join queues started appearing in literature approximately 30 years ago. Flatto and Hahn [24] considered a system with two service stations in parallel and exponential inter-arrival and exponential service times. They gave a parametric expression for the bi-variate distribution of the number in the second task queue when there are zero entities in the first task queue. In the sequel to this work, Flatto [23] analyzed the asymptotic interdependence between the number in the two queues as these numbers tend towards infinity. Flatto and Hahn [23]'s parametric expression was used by Nelson and Tantawi [47] to find the mean sojourn time in a system with two parallel tasks and exponential inter-arrival and service times. They also developed an approximation for the case of more than two parallel tasks using parameters found using simulations. Baccelli [4] derived the stability condition for the system with general inter-arrival and service times and obtained simple upper and lower bounds based on independence of the n queueing systems in the case of deterministic and independent arrivals respectively. Continuing this work, Baccelli [5]

used properties of associated random variables and stochastic orderings using given sub-sigma algebras to obtain bounds for the symmetric fork-join queueing system with general inter-arrival and service times. They also analyzed the asymptotics of the moments of the system response time as the number of parallel tasks grows to be very large. The performance of these bounds are not investigated in either of [6] or [5]. Kim and Agarwala [35] gave an approximation for a fork-join queueing system with only two tasks and general inter-arrival and service times. They provided a general form of the virtual waiting time in terms of unknown coefficients and these coefficients can be calculated recursively for each arriving entity. In steady state, these coefficients are approximated. For higher number of tasks, this representation becomes tedious and the number of coefficients might blow out of proportion. Varma and Makowski [61] considered systems with general inter-arrival and service times. They provided a conjecture for when the system is in a state of heavy traffic. Using this and light traffic approximations, they provided an interpolation approximation for any traffic intensity. Nelson et al [48] considered a different form of fork-join queue in which the number of tasks of each job is a discrete random variable. The tasks wait in a single queue and are served by the servers according to a FCFS service discipline. Tasks belonging to a the same job are served in a random order. They provided an iterative solution for this case. Towsley et al [58] extended this work to the case where the servers follow a processor sharing service discipline. They derived bounds and computed approximations for this case and compared the processor sharing service discipline against the FCFS service discipline. Their findings showed that systems operating under FCFS perform better in terms of response times. Due to lack of efficient approximations for these systems, Dai [16] and Chen et al [14] devised efficient simulation techniques for these systems. A review of the advances made in the response time estimation of fork-join queues until the year 1994 can be

found in the work of Boxma et al [11].

In spite of the high volume of research work in the area of fork-join queues, the hardness of the problem is evidenced by the fact that to the best of our knowledge, there is still no exact solution method for even the system with exponential inter-arrival and service times when the number of parallel tasks is greater than two. This is further reinforced in the Acknowledgement section of the work by Ko and Serfozo [36]. The only way to estimate the response time in fork-join queueing systems is simulations which require a huge amount of time and computing resources. In the current work, we present a numerical method to estimate the sojourn time in a fork join queueing system with exponential inter-arrival and general service times with no constraint on the number of tasks. To the best of our knowledge, no such technique exists in literature. Our results show that our method outperforms over all earlier methods significantly and in multiple respects.

We present a formal definition of the problem in the next section.

2.2 Problem Description

In the queueing system under consideration, jobs arrive according to a Poisson process with rate λ. On arrival, each job instantaneously splits into n tasks. There are n single server queueing stations with infinite buffer space and operating under FIFO service discipline. Task indexed by i, $1 \leq i \leq n$ is routed to the queue at queueing station i. We consider a symmetric (*i.e.* all servers and tasks are stochastically identical) fork-join queueing network. The service times for the tasks are independent and identically distributed across jobs and across constituting tasks of the same job. These service times are $\mathcal{R}_+$ valued and follow a general distribution with cumulative distribution function $G(x)$ that has a finite mean and variance. We denote the mean of $G(x)$ by $1/\mu$, *i.e.* μ is the average service rate. The traffic

intensity at each queueing station is denoted by $\rho = \frac{\lambda}{\mu}$. Each of the n task queueing stations has a join buffer. After completion of a task, it is routed to it's join buffer where the task waits for the rest of the tasks belonging to it's parent job to get completed. Joining is assumed to occur instantaneously. Therefore, the departure time of the last task of a job to finish service coincides with the departure time of the job from the system. This network model is shown in Figure 2.

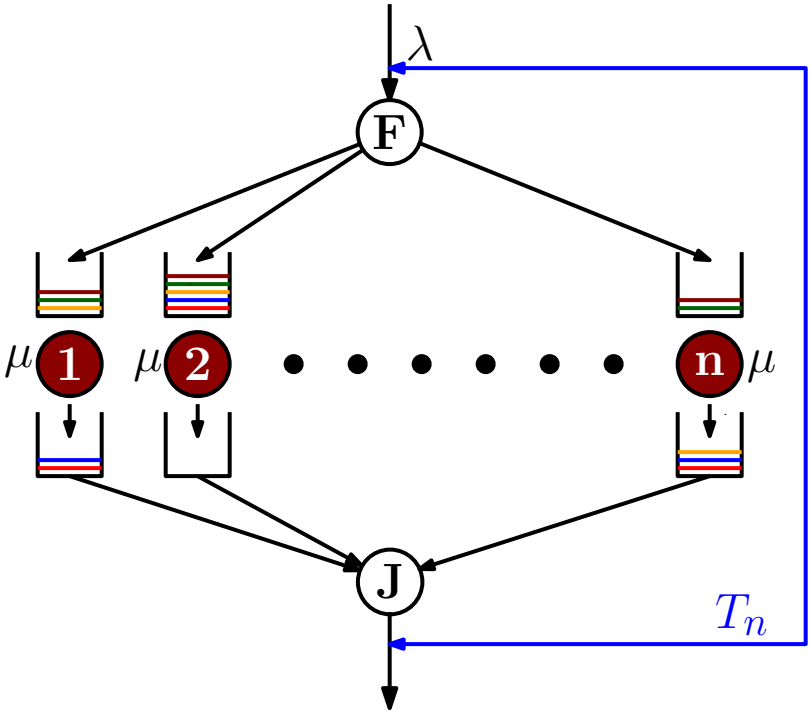

Figure 2: Symmetric n-Dimensional Fork-Join Queueing Network

Baccelli and Makowski [4] derived the stability condition for this system. This is same as the stability condition of a $G/G/1$ queue *i.e.* the symmetric fork-join queueing system is stable *iff* $\lambda < \mu$ which is same as $\rho < 1$.

Using index j, $j \geq 1$ for the arriving jobs in order of their arrival, we denote the

sequence of inter-arrival times of the jobs by $\{\tau_j,\ j = 1, 2, \ldots\}$ and the sequence of service times at task i, $1 \leq i \leq n$ by $\{\sigma_j^{(i)},\ j = 1, 2, \ldots\}$. We let $W_0^{(i)}$, defined by the initial state of the system, denote the initial workload in the queueing station of task i. We denote the workloads observed by subsequent customers by $W_j^{(i)}$. These are defined by the Lindley recursions as follows:

$$W_j^{(i)} = \left[W_{j-1}^{(i)} + \sigma_{j-1}^{(i)} - \tau_j\right]^+, \qquad i = 1, \ldots, n;\ j = 1, 2, \ldots$$

We denote the time interval between the arrival of job j and its completion of task i by $T_j^{(i)}$ which is defined below:

$$T_j^{(i)} = W_j^{(i)} + \sigma_j^{(i)}, \qquad i = 1, \ldots, n;\ j = 1, 2, \ldots$$

The response time of job j, denoted by $T_{j,n}$ is defined as:

$$T_{j,n} = \max_{i=1,\ldots,n} T_j^{(i)} \qquad j = 1, 2, \ldots \tag{1}$$

We denote the response time of any job in steady state by T_n. When $\rho < 1$, the steady state distribution of T_n exists, has a finite mean represented by $E[T_n]$, and the distribution is independent of the initial state of the system. This $E[T_n]$ is the system performance measure that is of interest to us. It is to be noted that the assumption of instantaneous join operation of the tasks does not impact the approximation derived in this work, and can be removed by adding the average joining time to the estimated value of $E[T_n]$.

From Equation (1) it is seen that T_n is the maximum of n identically distributed random variables. Each of these n random variables is the response time of an $M/G/1$ queueing system whose steady state distribution is known. However, there

are no known methods for the evaluation of $E[T_n]$ using Equation (1) because these n random variables are not independent. The synchronized arrivals to all the n task queues introduce a correlation between the queue lengths. This correlation has been found to be extremely hard to quantify thus proving a closed form expression for $E[T_n]$ elusive for many researchers. In the next section we present an estimation technique for $E[T_n]$. In the rest of this dissertation, for brevity $E[T_n]$ is denoted by $\overline{T}_n$. Table 1 lists the notations used.

Notation	Description
λ	Mean arrival rate
μ	Mean service rate for each task
$G(x)$	CDF of service times
ρ	Traffic intensity ($= \frac{\lambda}{\mu}$)
T_n	Random variable representing steady state response time
$\overline{T}_n$	Expected steady state response time

Table 1: List of Notations

2.3 Response Time Estimation

We present an approximation technique whose performance is excellent in terms of the error percentage observed with respect to the simulated value of the expected response time. The conjecture which forms the basis of this technique is the following:

Conjecture 1. *The mean response time of the symmetric n-dimensional fork-join queueing network is linear with respect to* $\frac{\rho}{1-\rho}$*, i.e. the slope and intercept are independent of the traffic intensity,* ρ*.*

$$\overline{T}_n = \frac{\rho m_n}{\mu(1-\rho)} + M_n \tag{2}$$

where, m_n is a parameter independent of ρ and $M_n = \int_0^\infty nx\{G(x)\}^{n-1}dG(x)$.

The constant term in Equation (2) is the maximum of n *i.i.d.* random variables with CDF $G(x)$. This is explained by the following reasoning: as ρ approaches zero from the right, an arriving job sees an empty system with probability approaching approaching one *i.e.* an arriving entity will find an empty system almost surely. Therefore, with probability approaching one, the response time of any job arriving into the system is the maximum of the service times at the individual tasks and since these times are independent, we obtain M_n in Equation (2).

We do not have an analytical expression for the parameter m_n currently and it needs to be estimated using simulations.

Remark 1. *The result in Conjecture 1 is satisfied by systems with general service times when $n = 1$ and exponential service times when $n = 2$, for which the analytical expressions for average response time are available. When $n = 1$, the system is an $M/G/1$ queue. The expected response time in steady state is $\frac{(1+C_v^2)\rho}{2\mu(1-\rho)} + \frac{1}{\mu}$. In this case, the parameter $m_1 = \frac{1+C_v^2}{2}$, where C_v^2 is the squared coefficient of variation of the service time distribution $G(x)$. When $n = 2$ and service times are exponentially distributed, Nelson and Tantawi [47] show that the expected response time in steady state is $\frac{12-\rho}{8\mu(1-\rho)}$ which can be written as $\frac{11\rho}{8\mu(1-\rho)} + \frac{3}{2\mu}$. In this case $m_2 = \frac{11}{8}$ and $M_2 = \frac{3}{2}$.*

Intuition behind Conjecture 1. The expected response time when the number of tasks is n can be written in the following form:

$$E[T_n] = E[T_{n-1}] + E[T^{(n)} - T_{n-1}|T^{(n)} > T_{n-1}]P(T^{(n)} > T_{n-1}) \tag{3}$$

Here we revert to the traditional representation of expectations for clarity. $T^{(n)}$ denotes the steady state response time random variable of the nth task. $P(T^{(n)} >$

T_{n-1}) is the probability that the nth task is the last one to get completed among the n tasks. Due to the symmetry of the system, we have the following relationship:

$$P(T^{(n)} > T_{n-1}) = \frac{1}{n} \tag{4}$$

Using an induction argument, since Conjecture 1 holds when the number of tasks $n = 1$, we assume that Conjecture 1 holds when the number of tasks is $n - 1$ and obtain the following relationship:

$$E[T_n] = \frac{\rho m_{n-1}}{\mu(1-\rho)} + M_{n-1} + \frac{E[T^{(n)} - T_{n-1} | T^{(n)} > T_{n-1}]}{n} \tag{5}$$

$E[T^{(n)} - T_{n-1} | T^{(n)} > T_{n-1}]$ is the expected excess time needed to complete the nth task given that it is the last one to get completed. The expected workloads of each of the individual $M/G/1$ queues are known to be linear with respect to $\frac{\rho}{1-\rho}$. As ρ increases, $\frac{\rho}{1-\rho}$ increases and the variance of the individual workloads increases. Therefore, it is expected that the excess time needed to complete the last task also increases with increase in $\frac{\rho}{1-\rho}$. When these quantities were plotted, they displayed a very convincing linear relationship.

To demonstrate the performance of Conjecture 1, we plot the average response time obtained using simulations against $\frac{\rho}{1-\rho}$ for the case of $n = 5$, $\mu = 1$ and exponential service time distribution. The linear relationship is observed in Figure 3.

When the task service times are exponentially distributed, the parameter m_n in Equation (2) displays some special properties which are explained in the next subsection.

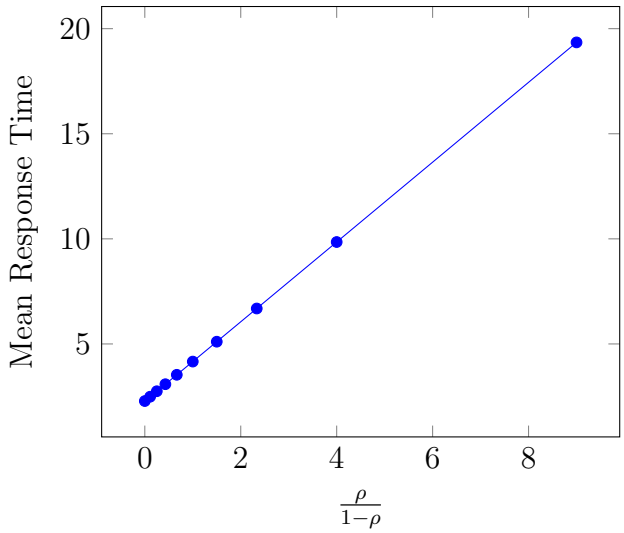

Figure 3: Plot of simulated average response time vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $\mu = 1$ and exponential service time distribution

2.3.1 Exponential Service Times

When the service times are exponential, we have $G(x) = 1 - e^{-\mu x}$. In this case, $M_n = \frac{H_n}{\mu}$, where, $H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$, is the harmonic sum. Furthermore, the parameter m_n is observed to have two interesting properties:

- m_n is independent of the service rate μ
- m_n is directly proportional to the natural logarithm of n. This linear relationship is demonstrated in Fig. 4.

As mentioned in Remark 1, the values of m_n for $n = 1$ and $n = 2$ are known. Using these values and the observed properties, we arrive at the following conjecture:

Conjecture 2. *The mean response time of the symmetric n-dimensional symmetric fork-join queueing network with exponential inter-arrival and service times is given by the following relationship:*

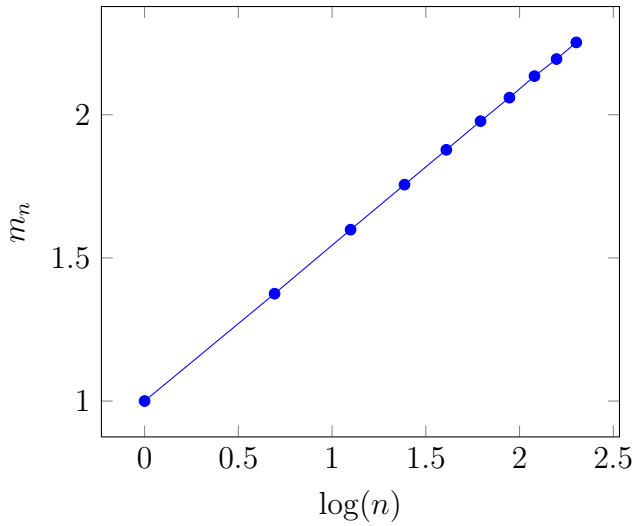

Figure 4: Plot of parameter m_n vs. $\log(n)$

$$\overline{T}_n = \frac{\rho\,[0.541\,\log(n) + 1]}{\mu(1-\rho)} + \frac{H_n}{\mu} \tag{6}$$

Intuition behind Conjecture 2. A lower bound for $\overline{T}_n$ is given by the expectation of the maximum of n *i.i.d.* task service time random variables *i.e.* $\frac{H_n}{\mu}$. Nelson and Tantawi show that an upper bound is given by the expectation of the maximum of n *i.i.d.* $M/M/1$ response time random variables. Since the response time of an $M/M/1$ queueing system is also an exponential random variable with mean $\frac{1}{\mu(1-\rho)}$, the upper bound is given by $\frac{H_n}{\mu(1-\rho)}$. Considering that both the lower bound (which is also the constant term in Equation 2) and the upper bound increase linearly with H_n, the first impulse is to check if m_n also increases linearly with respect to H_n. However, experiments show that this does not hold. Nevertheless, since H_n increases logarithmically with n, therefore, the values of m_n obtained using simulations were plotted against $\log(n)$ and the linear fit demonstrated in Figure 4 was observed.

Even though we have been unable to prove Conjectures 1 and 2, we have conducted extensive experiments to establish their utility as a tool for approximating the expected response time of the symmetric fork-join queueing system. In the next section, an algorithm constructed using Conjecture 1 for the computation of the expected response time is presented.

2.4 Response Time Computation Algorithm for Non-Exponential Service Times

Given the number of parallel tasks n, the mean service rate μ and the service time CDF $G(x)$, $\overline{T}_n$ only depends on the traffic intensity ρ. *For clarity, we write this as* $\overline{T}_n(\rho)$. When the task service times are exponentially distributed, the estimate for the mean response time is obtained directly from Equation 6. When, the task service times are not exponential, to compute the response time for any traffic intensity ρ using Equation (2), the value of m_n needs to be estimated. Towards this end, the system is simulated to estimate the mean response time for one value of ρ, say $\rho = 0.5$. We denote this estimate obtained using simulation by $\widehat{\overline{T}}_5^{sim}(0.5)$. Using this estimate, Equation (2) is now solved to obtain an estimate of m_n represented by $\widehat{m}_n$. Since according to Conjecture 1, m_n is independent of ρ, the estimate $\widehat{m}_n$ can now be used to estimate the mean response time for any other value of ρ in the interval $[0, 1)$. *Therefore, by simulating the system for one value of the traffic intensity* ρ*, it has become possible to obtain an estimate of the mean response time for infinitely many values of* ρ*.* This technique is explained formally in Algorithm 1. *We denote the estimate of* $\overline{T}_n$ *obtained using Equation 6 for exponential task service times or by the method outlined above for general service times by* $\widehat{\overline{T}}_n$.

We now demonstrate the use of Conjecture 2 and Algorithm 1 to estimate the mean response time using numerical examples.

Algorithm 1 Computation of $\widehat{\overline{T}}_n$

Input: Number of parallel tasks, n; Service rate, μ; Service time distribution $G(.)$
Output: Expression for expected sojourn time for any traffic intensity, $\rho \in [0, 1)$

1: Simulate the system for $\rho = 0.5$. Estimate expected sojourn time $\widehat{\overline{T}}_5^{sim}(0.5)$.
2: Compute $M_n = \int_0^\infty nx\{G(x)\}^{n-1}dG(x)$
3: Solve linear equation for $\widehat{m}_n$: $\widehat{\overline{T}}_5^{sim}(0.5) = \frac{0.5\widehat{m}_n}{\mu(1-0.5)} + M_n$
4: **return** $\widehat{\overline{T}}_n(\rho) = \frac{\rho\widehat{m}_n}{\mu(1-\rho)} + M_n$

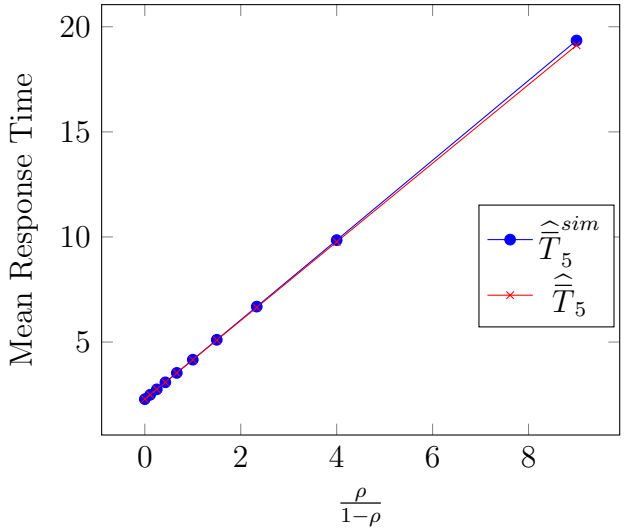

Figure 5: Plot of simulated and predicted average response times vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $\mu = 1$ and exponential service time distribution

2.4.1 Numerical Example

Consider a fork-join queueing system with $n = 5$, $\mu = 1$ and exponential service time distribution. Using Conjecture 2, we obtain the following approximation: $\widehat{\overline{T}}_5 = \frac{\rho\,[0.541\,\log(5)+1]}{1(1-\rho)} + \frac{2.283}{1}$. We plot this approximation and compare it against the values obtained using simulations for ρ values of 0.1 to 0.9 at increments of 0.1 in Figure 5. The closeness of the approximation is obvious in the figure.

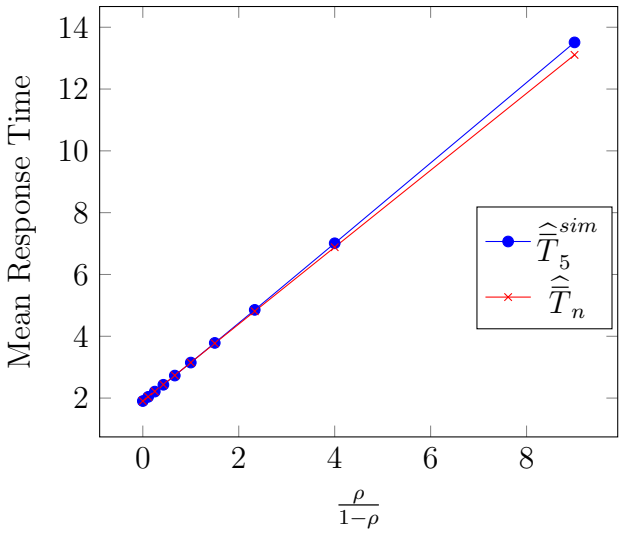

Figure 6: Plot of simulated and predicted average response times vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $\mu = 1$ and Erlang-2 service time distribution

Now consider a system with $n = 5$, $\mu = 1$ and Erlang-2 service time distribution. Equation (2) results in the following linear relationship: $\overline{T}_5(\rho) = \frac{\rho m_5}{(1-\rho)} + 1.904$. The value of $E[T_5](\rho)$ for only one value of ρ is needed to find the the unknown parameter, m_5. Therefore, the system is simulated for $\rho = 0.5$ and the value of $\widehat{\overline{T}}_5^{sim}(0.5)$ is estimated to be equal to 3.149. The simulation parameters used for obtaining the estimate of $\overline{T}_n$ are detailed in Section 2.5. The following linear equation is now solved for $\widehat{m}_n$: $3.149 = \widehat{m}_n + 1.904$ which gives $\widehat{m}_n = 1.245$. Thus, the value of m_5 has been estimated for estimating the value of $\overline{T}_5$ for any other value of the traffic intensity ρ. This is given by: $\widehat{\overline{T}}_n(\rho) = \frac{1.245\rho}{\mu(1-\rho)} + 1.904$. Figure 6 shows the plot of the simulated average response times for ρ values of 0.1 to 0.9 at increments of 0.1 and the corresponding predicted value using the simulation for $\rho = 0.5$ and $\mu = 1$. The closeness of the approximation to the simulated values leads us to believe that our conjecture is indeed correct.

In the next section, we describe our experimental results on the response-time

estimation of the symmetric fork-join queueing system and compare against existing approximation techniques in the literature.

2.5 Results and Comparisons

Considering the hardness of the problem, exact estimation techniques for the symmetric fork-join queueing system do not exist. In this section, we compare the results obtained using our approximation technique with those obtained using simulations. Furthermore, we describe existing approximation methods in the literature and compare those against our method. We present our arguments based on experiments conducted with different values of the number of tasks n and for different service time distributions.

For exponential service time distribution, results are reported for $n = 3$, 5, 10, 20, 30, 40 and 50 (Tables (2)–(8)). For Erlang-2 service time distribution, results are reported for $n = 3$, 5, 10, 15, 20 and 30 (Tables (9)–(14)). Lastly for Pareto service time distribution, results are reported for $n = 2$, 3, 5, 7 and 10 (Tables (15)–(19)). The mean service rate μ for all simulations and calculations was assumed to take unit value, *i.e.* $\mu = 1$. This determines the parameters for the exponential and Erlang-2 distributions. The Pareto distribution parameters are $\alpha = 2.291$ and $x_m = 0.563$.

The values of n for reporting results were chosen based on the reliability of simulations with respect to reaching steady state and having reasonable variance. For exponential and Erlang-2 service time distributions, results are reported for the traffic intensity ρ ranging from 0.1 to 0.9. For the Pareto service time distribution results, since the service time variability is higher, reasonable values could only be obtained for up till $\rho = 0.8$, except for the case of $n = 2$ for which the results for $\rho = 0.9$ are also reported.

In Tables (2)–(19), for different values of n, $\widehat{\overline{T}}_n^{sim}$ denotes the estimate of the av-

Table 2: Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 3$ and exponential service time distribution

ρ	$\widehat{\widehat{T}}_3^{sim}$	$\widehat{\widehat{T}}_3$	% Error	$\widehat{\widehat{T}}_3^{NT}$	% Error	$\widehat{\widehat{T}}_3^{VM}$	% Error	$\widehat{\widehat{T}}_3^{KS}$	% Error
0.1	2.01±0.000	2.01	**-0.002**	2.01	0.180	2.011	0.009	2.21	10.131
0.2	2.23±0.000	2.23	-0.020	2.22	0.373	2.232	**-0.005**	2.49	11.586
0.3	2.51±0.000	2.52	-0.050	2.50	0.573	2.517	**-0.034**	2.85	13.065
0.4	2.89±0.001	2.90	-0.0921	2.88	0.780	2.896	**-0.072**	3.32	14.573
0.5	3.43±0.001	3.43	**-0.141**	3.40	1.014	3.428	-0.146	3.99	16.114
0.6	4.23±0.001	4.22	-0.207	4.18	1.218	4.225	**-0.201**	4.98	17.677
0.7	5.57±0.003	5.55	-0.307	5.49	1.459	5.553	**-0.298**	6.64	19.245
0.8	8.24±0.007	8.21	-0.453	8.10	1.694	8.210	**-0.397**	9.96	20.803
0.9	16.27±0.025	16.18	**-0.582**	15.95	2.122	16.181	-0.703	19.93	22.429

Table 3: Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 5$ and exponential service time distribution

ρ	$\widehat{\widehat{T}}_5^{sim}$	$\widehat{\widehat{T}}_5$	% Error	$\widehat{\widehat{T}}_5^{NT}$	% Error	$\widehat{\widehat{T}}_5^{VM}$	% Error	$\widehat{\widehat{T}}_5^{KS}$	% Error
0.1	2.49±0.000	2.49	0.047	2.50	0.503	2.49	**-0.001**	2.97	19.365
0.2	2.75±0.000	2.75	0.051	2.78	0.989	2.75	**-0.044**	3.34	21.607
0.3	3.08±0.000	3.09	**0.024**	3.13	1.471	3.08	-0.121	3.82	23.897
0.4	3.53±0.001	3.53	**-0.045**	3.60	1.940	3.52	-0.243	4.46	26.223
0.5	4.16±0.001	4.15	**-0.164**	4.26	2.388	4.14	-0.418	5.35	28.578
0.6	5.11±0.002	5.09	**-0.343**	5.25	2.805	5.07	-0.653	6.69	30.949
0.7	6.69±0.003	6.65	**-0.568**	6.90	3.207	6.62	-0.935	8.92	33.356
0.8	9.85±0.009	9.77	**-0.845**	10.20	3.590	9.72	-1.273	13.37	35.793
0.9	19.35±0.033	19.12	**-1.176**	20.11	3.950	19.03	-1.666	26.75	38.260

Table 4: Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 10$ and exponential service time distribution

ρ	$\widehat{\bar{T}}_{10}^{sim}$	$\widehat{\bar{T}}_{10}$	% Error	$\widehat{\bar{T}}_{10}^{NT}$	% Error	$\widehat{\bar{T}}_{10}^{VM}$	% Error	$\widehat{\bar{T}}_{10}^{KS}$	% Error
0.1	3.17±0.000	3.18	0.143	3.21	1.259	3.17	**-0.031**	4.06	27.906
0.2	3.48±0.000	3.49	0.226	3.57	2.523	3.48	**-0.127**	4.57	31.145
0.3	3.88±0.001	3.89	**0.236**	4.03	3.780	3.87	-0.305	5.22	34.448
0.4	4.42±0.001	4.43	**0.161**	4.64	5.019	4.39	-0.579	6.09	37.804
0.5	5.18±0.002	5.17	**-0.022**	5.50	6.219	5.13	-0.970	7.31	41.185
0.6	6.32±0.002	6.30	**-0.323**	6.78	7.370	6.22	-1.488	9.13	44.577
0.7	8.23±0.005	8.17	**-0.751**	8.93	8.463	8.05	-2.143	12.18	47.970
0.8	12.07±0.014	11.91	**-1.348**	13.22	9.453	11.72	-2.975	18.27	51.298
0.9	23.64±0.063	23.14	**-2.134**	26.08	10.314	22.70	-4.003	36.54	54.524

Table 5: Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 20$ and exponential service time distribution

ρ	$\widehat{\bar{T}}_{20}^{sim}$	$\widehat{\bar{T}}_{20}$	% Error	$\widehat{\bar{T}}_{20}^{NT}$	% Error	$\widehat{\bar{T}}_{20}^{VM}$	% Error	$\widehat{\bar{T}}_{20}^{KS}$	% Error
0.1	3.88±0.000	3.89	0.289	3.95	1.885	3.88	**-0.047**	5.19	33.743
0.2	4.23±0.001	4.25	0.490	4.39	3.786	4.22	**-0.203**	5.83	37.858
0.3	4.69±0.001	4.72	0.578	4.96	5.683	4.67	**-0.491**	6.67	42.059
0.4	5.32±0.001	5.34	**0.533**	5.72	7.555	5.27	-0.938	7.78	46.321
0.5	6.20±0.002	6.22	**0.326**	6.78	9.378	6.10	-1.566	9.34	50.608
0.6	7.53±0.003	7.53	**-0.072**	8.37	11.118	7.35	-2.405	11.67	54.878
0.7	9.78±0.005	9.71	**-0.698**	11.03	12.734	9.44	-3.494	15.56	59.067
0.8	14.31±0.010	14.08	**-1.604**	16.34	14.163	13.61	-4.881	23.34	63.084
0.9	27.96±0.040	27.18	**-2.787**	32.27	15.400	26.13	-6.560	46.68	66.915

Table 6: Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 30$ and exponential service time distribution

ρ	$\widehat{\bar{T}}_{30}^{sim}$	$\widehat{\bar{T}}_{30}$	% Error	$\widehat{\bar{T}}_{30}^{NT}$	% Error	$\widehat{\bar{T}}_{30}^{VM}$	% Error	$\widehat{\bar{T}}_{30}^{KS}$	% Error
0.1	4.29±0.000	4.31	0.381	4.39	2.197	4.29	**-0.058**	5.86	36.354
0.2	4.67±0.001	4.71	0.659	4.88	4.418	4.66	**-0.249**	6.59	40.926
0.3	5.17±0.001	5.21	0.805	5.51	6.639	5.14	**-0.600**	7.53	45.598
0.4	5.84±0.001	5.89	**0.791**	6.36	8.834	5.78	-1.144	8.78	50.337
0.5	6.79±0.002	6.84	0.**586**	7.54	10.973	6.67	-1.911	10.54	55.101
0.6	8.24±0.003	8.26	**0.153**	9.32	13.016	8.00	-2.935	13.17	59.834
0.7	10.68±0.005	10.62	**-0.544**	12.27	14.921	10.23	-4.251	17.57	64.475
0.8	15.59±0.013	15.36	**-1.535**	18.19	16.646	14.68	-5.887	26.35	68.959
0.9	30.39±0.043	29.56	**-2.755**	35.94	18.266	28.03	-7.779	52.70	73.387

Table 7: Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 40$ and exponential service time distribution

ρ	$\widehat{\bar{T}}_{40}^{sim}$	$\widehat{\bar{T}}_{40}$	% Error	$\widehat{\bar{T}}_{40}^{NT}$	% Error	$\widehat{\bar{T}}_{40}^{VM}$	% Error	$\widehat{\bar{T}}_{40}^{KS}$	% Error
0.1	4.59±0.000	4.61	0.449	4.70	2.400	4.59	**-0.065**	6.33	37.948
0.2	4.99±0.000	5.03	0.782	5.23	4.827	4.97	**-0.279**	7.12	42.820
0.3	5.51±0.001	5.56	0.971	5.91	7.257	5.47	**-0.678**	8.14	47.801
0.4	6.21±0.001	6.28	**0.978**	6.82	9.657	6.13	-1.295	9.50	52.847
0.5	7.22±0.001	7.27	**0.771**	8.08	11.994	7.06	-2.165	11.40	57.914
0.6	8.74±0.002	8.77	**0.312**	9.99	14.225	8.45	-3.324	14.25	62.942
0.7	11.32±0.004	11.27	**-0.446**	13.16	16.296	10.77	-4.816	19.00	67.847
0.8	16.51±0.011	16.26	**-1.510**	19.51	18.190	15.41	-6.646	28.50	72.602
0.9	32.19±0.069	31.24	**-2.959**	38.57	19.802	29.33	-8.886	57.00	77.047

Table 8: Comparison with simulation results and approximations of Nelson and Tantawi [47], Varma and Makowski [61] and Ko and Serfozo [36] for $n = 50$ and exponential service time distribution

ρ	$\widehat{\widehat{T}}_{50}^{sim}$	$\widehat{\widehat{T}}_{50}$	% Error	$\widehat{\widehat{T}}_{50}^{NT}$	% Error	$\widehat{\widehat{T}}_{50}^{VM}$	% Error	$\widehat{\widehat{T}}_{50}^{KS}$	% Error
0.1	4.82±0.000	4.84	0.500	4.94	2.545	4.82	**-0.062**	6.70	39.059
0.2	5.23±0.000	5.28	0.877	5.50	5.123	5.22	**-0.289**	7.54	44.152
0.3	5.77±0.001	5.83	1.100	6.22	7.707	5.73	**-0.713**	8.62	49.362
0.4	6.50±0.001	6.58	**1.130**	7.17	10.260	6.41	-1.373	10.06	54.640
0.5	7.55±0.002	7.62	**0.921**	8.51	12.738	7.37	-2.314	12.07	59.926
0.6	9.13±0.003	9.17	**0.440**	10.51	15.100	8.81	-3.570	15.09	65.160
0.7	11.81±0.006	11.77	**-0.351**	13.86	17.302	11.20	-5.174	20.11	70.276
0.8	17.22±0.016	16.96	**-1.471**	20.54	19.314	15.99	-7.142	30.17	75.225
0.9	33.56±0.086	32.55	**-3.008**	40.61	21.015	30.35	-9.556	60.34	79.818

Table 9: Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 3$ and Erlang-2 service time distribution

ρ	$\widehat{\widehat{T}}_{3}^{sim}$	$\widehat{\widehat{T}}_{3}$	% Error	$\widehat{\widehat{T}}_{3}^{TT}$	% Error	$\widehat{\widehat{T}}_{3}^{VM}$	% Error
0.1	1.73±0.000	1.73	0.100	1.93	11.658	1.73	**0.003**
0.2	1.88±0.000	1.88	0.163	2.03	7.966	1.88	**-0.037**
0.3	2.07±0.000	2.07	0.176	2.17	4.566	2.07	**-0.130**
0.4	2.33±0.000	2.33	**0.127**	2.37	1.463	2.32	-0.297
0.6	3.25±0.001	3.24	**-0.240**	3.13	-3.854	3.22	-0.920
0.7	4.18±0.003	4.15	**-0.622**	3.93	-6.080	4.12	-1.445
0.8	6.04±0.007	5.97	**-1.208**	5.56	-8.051	5.91	-2.184
0.9	11.67±0.031	11.43	**-2.041**	10.53	-9.778	11.30	-3.179

Table 10: Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 5$ and Erlang-2 service time distribution

ρ	$\widehat{\overline{T}}_5^{sim}$	$\widehat{\overline{T}}_5$	% Error	$\widehat{\overline{T}}_5^{TT}$	% Error	$\widehat{\overline{T}}_5^{VM}$	% Error
0.1	2.04±0.000	2.04	0.154	2.33	14.013	2.04	**0.002**
0.2	2.21±0.000	2.22	0.251	2.40	8.675	2.21	**-0.064**
0.3	2.43±0.000	2.44	0.274	2.52	3.655	2.43	**-0.220**
0.4	2.73±0.000	2.73	**0.200**	2.70	-1.051	2.72	-0.486
0.6	3.78±0.001	3.77	**-0.371**	3.42	-9.552	3.73	-1.483
0.7	4.85±0.003	4.81	**-0.954**	4.21	-13.372	4.74	-2.303
0.8	7.01±0.006	6.88	**-1.809**	5.82	-16.929	6.77	-3.411
0.9	13.51±0.028	13.11	**-2.992**	10.78	-20.238	12.85	-4.863

Table 11: Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 10$ and Erlang-2 service time distribution

ρ	$\widehat{\overline{T}}_{10}^{sim}$	$\widehat{\overline{T}}_{10}$	% Error	$\widehat{\overline{T}}_{10}^{TT}$	% Error	$\widehat{\overline{T}}_{10}^{VM}$	% Error
0.1	2.47±0.000	2.47	0.228	2.87	16.245	2.47	**-0.030**
0.2	2.66±0.000	2.67	0.369	2.91	9.325	2.66	**-0.171**
0.3	2.92±0.000	2.93	**0.402**	3.00	2.741	2.91	-0.440
0.4	3.26±0.001	3.27	**0.293**	3.15	-3.517	3.24	-0.879
0.6	4.50±0.001	4.48	**-0.539**	3.82	-15.119	4.39	-2.448
0.7	5.76±0.002	5.68	**-1.390**	4.58	-20.509	5.55	-3.711
0.8	8.31±0.006	8.09	**-2.631**	6.18	-25.659	7.86	-5.391
0.9	16.00±0.037	15.31	**-4.330**	11.11	-30.584	14.79	-7.553

Table 12: Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 15$ and Erlang-2 service time distribution

ρ	$\widehat{\overline{T}}_{15}^{sim}$	$\widehat{\overline{T}}_{15}$	% Error	$\widehat{\overline{T}}_{15}^{TT}$	% Error	$\widehat{\overline{T}}_{15}^{VM}$	% Error
0.1	2.71±0.000	2.72	0.266	3.18	17.210	2.71	**-0.062**
0.2	2.93±0.000	2.94	0.436	3.21	9.598	2.92	**-0.250**
0.3	3.20±0.000	3.22	**0.477**	3.28	2.325	3.18	-0.602
0.4	3.57±0.000	3.59	**0.349**	3.41	-4.622	3.53	-1.152
0.6	4.92±0.001	4.88	**-0.626**	4.05	-17.609	4.76	-3.085
0.7	6.28±0.002	6.18	**-1.598**	4.79	-23.685	5.99	-4.589
0.8	9.05±0.007	8.78	**-2.979**	6.38	-29.503	8.46	-6.541
0.9	17.42±0.026	16.56	**-4.909**	11.30	-35.127	15.84	-9.072

Table 13: Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 20$ and Erlang-2 service time distribution

ρ	$\widehat{\overline{T}}_{20}^{sim}$	$\widehat{\overline{T}}_{20}$	% Error	$\widehat{\overline{T}}_{20}^{TT}$	% Error	$\widehat{\overline{T}}_{20}^{VM}$	% Error
0.1	2.89±0.000	2.90	0.289	3.41	17.791	2.89	**-0.091**
0.2	3.11±0.000	3.13	0.480	3.42	9.766	3.10	**-0.318**
0.3	3.40±0.000	3.42	**0.529**	3.47	2.080	3.38	-0.723
0.4	3.79±0.001	3.81	**0.386**	3.59	-5.286	3.74	-1.362
0.6	5.21±0.002	5.17	**-0.697**	4.21	-19.125	5.02	-3.561
0.7	6.65±0.004	6.53	**-1.778**	4.95	-25.642	6.30	-5.266
0.8	9.58±0.010	9.26	**-3.339**	6.52	-31.928	8.86	-7.491
0.9	18.44±0.043	17.44	**-5.431**	11.43	-37.981	16.54	-10.285

Table 14: Comparison with simulation results and approximations of Thomasian and Tantawi [47] and Varma and Makowski [61] for $n = 30$ and Erlang-2 service time distribution

ρ	$\widehat{\overline{T}}_{30}^{sim}$	$\widehat{\overline{T}}_{30}$	% Error	$\widehat{\overline{T}}_{30}^{TT}$	% Error	$\widehat{\overline{T}}_{30}^{VM}$	% Error
0.1	3.14±0.000	3.15	0.328	3.72	18.522	3.13	**-0.126**
0.2	3.37±0.000	3.39	0.539	3.71	10.002	3.36	**-0.410**
0.3	3.68±0.000	3.70	**0.592**	3.74	1.819	3.64	-0.900
0.4	4.10±0.001	4.11	**0.432**	3.85	-6.049	4.03	-1.651
0.6	5.61±0.003	5.57	**-0.776**	4.44	-20.916	5.37	-4.198
0.7	7.16±0.006	7.02	**-1.984**	5.16	-27.964	6.72	-6.155
0.8	10.30±0.014	9.92	**-3.714**	6.72	-34.785	9.41	-8.682
0.9	19.83±0.051	18.63	**-6.048**	11.62	-41.392	17.48	-11.856

Table 15: Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 2$ and Pareto service time distribution

ρ	$\widehat{\overline{T}}_{2}^{sim}$	$\widehat{\overline{T}}_{2}$	% Error	$\widehat{\overline{T}}_{2}^{VM}$	% Error
0.1	1.50±0.011	1.49	**-0.134**	1.48	-1.297
0.2	1.77±0.026	1.76	**-0.221**	1.72	-2.437
0.3	2.11±0.045	2.11	**-0.250**	2.04	-3.422
0.4	2.57±0.069	2.57	**-0.159**	2.46	-4.214
0.6	4.17±0.156	4.18	**0.247**	3.94	-5.383
0.7	5.75±0.244	5.79	**0.653**	5.42	-5.693
0.8	8.87±0.424	9.00	**1.519**	8.38	-5.530
0.9	18.10±0.982	18.6609	**3.111**	17.25	-4.663

Table 16: Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 3$ and Pareto service time distribution

ρ	$\widehat{\overline{T}}_3^{sim}$	$\widehat{\overline{T}}_3$	% Error	$\widehat{\overline{T}}_3^{VM}$	% Error
0.1	1.82±0.052	1.81	**-0.386**	1.73	-4.860
0.2	2.21±0.114	2.20	**-0.543**	2.02	-8.808
0.3	2.72±0.195	2.71	**-0.567**	2.39	-12.097
0.4	3.39±0.304	3.38	**-0.401**	2.89	-14.795
0.6	5.69±0.681	5.73	**0.683**	4.63	-18.620
0.7	7.96±1.076	8.08	**1.510**	6.37	-19.956
0.8	12.42±1.881	12.78	**2.854**	9.85	-20.716

Table 17: Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 5$ and Pareto service time distribution

ρ	$\widehat{\overline{T}}_5^{sim}$	$\widehat{\overline{T}}_5$	% Error	$\widehat{\overline{T}}_5^{VM}$	% Error
0.1	2.30±0.051	2.28	**-0.648**	2.12	-7.936
0.2	2.86±0.112	2.84	**-0.935**	2.46	-14.096
0.3	3.58±0.192	3.55	**-0.980**	2.90	-19.010
0.4	4.53±0.297	4.50	**-0.702**	3.49	-22.894
0.6	7.73±0.658	7.81	**1.034**	5.55	-28.197
0.7	10.85±1.038	11.13	**2.602**	7.62	-29.810
0.8	16.87±1.792	17.77	**5.307**	11.74	-30.422

Table 18: Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 7$ and Pareto service time distribution

ρ	$\widehat{\overline{T}}_7^{sim}$	$\widehat{\overline{T}}_7$	% Error	$\widehat{\overline{T}}_7^{VM}$	% Error
0.1	2.66±0.033	2.63	**-0.918**	2.42	-8.977
0.2	3.33±0.074	3.28	**-1.371**	2.80	-15.868
0.3	4.17±0.129	4.11	**-1.405**	3.28	-21.229
0.4	5.27±0.199	5.22	**-0.953**	3.93	-25.359
0.6	8.95±0.439	9.09	**1.554**	6.20	-30.753
0.7	12.46±0.679	12.97	**4.049**	8.47	-32.055
0.8	19.16±1.176	20.72	**8.105**	13.00	-32.145

Table 19: Comparison with simulation results and approximation of Varma and Makowski [61] for $n = 10$ and Pareto service time distribution

ρ	$\widehat{\overline{T}}_{10}^{sim}$	$\widehat{\overline{T}}_{10}$	% Error	$\widehat{\overline{T}}_{10}^{VM}$	% Error
0.1	3.23±0.075	3.18	**-1.383**	2.79	-13.546
0.2	4.17±0.162	4.09	**-1.887**	3.21	-23.091
0.3	5.35±0.274	5.25	**-1.798**	3.74	-30.130
0.4	6.88±0.425	6.80	**-1.194**	4.44	-35.431
0.6	12.00±0.957	12.23	**1.928**	6.92	-42.278
0.7	16.83±1.491	17.65	**4.891**	9.40	-44.124
0.8	26.00±2.577	28.50	**9.643**	14.36	-44.752

erage response time obtained using simulations, $\widehat{\overline{T}}_n$ denotes the approximation using the methods described in Section 2.3, $\widehat{\overline{T}}_n^{NT}$ denotes the approximation by Nelson and Tantawi [47], $\widehat{\overline{T}}_n^{TT}$ denotes the approximation by Thomasian and Tantawi [57], $\widehat{\overline{T}}_n^{VM}$ describes the approximation by Varma and Makowski [61] and $\widehat{\overline{T}}_n^{KS}$ denotes the approximation by Ko and Serfozo [36]. The approximations by these authors will be described briefly in Sections (2.5.2)–(2.5.4).

For all approximation techniques considered, we calculate the error percentage based on the mean response time from simulations and list them in the column next to the respective approximation. The error percentage calculation is the same for all the approximation methods compared. For example, for the techniques described in Section 2.3, the error percentage is calculated as follows:

$$\%Error = \frac{\left(\widehat{\overline{T}}_n - \widehat{\overline{T}}_n^{sim}\right) \times 100}{\widehat{\overline{T}}_n^{sim}} \tag{7}$$

The percentage error that is the least among the approximations compared is shown in bold font in Tables (2)–(19).

We now describe how our approximation compares with the response times estimated using simulations and other approximations in literature.

2.5.1 Comparison with Simulations

Simulations were run on a DELL OPTIPLEX 960 computer with INTEL(R) CORE(TM) 2 QUAD 3 GHZ processor and 8 GB of RAM. The simulations were run using Microsoft Visual Studio 2010. Each simulation result reported is a result of 10 simulation runs. For each simulation run, 30 million arriving job entities and the corresponding n service times were generated. The response time of each entity was calculated using Equation 1. Out of these, response times of the first 10 million entities was thrown out to account for warm up period. Response times of the last 20 million entities were used for the calculation of the average response time. 95% confidence intervals are reported for each average response time. These confidence intervals are generated using the average response times in the 10 simulation runs.

For the case of exponential service times (Tables (2)–(8)), the estimate is obtained directly using Conjecture 2 and no simulation is necessary. The estimated mean response time $\widehat{\widetilde{T}}_n$ is remarkably close to the values observed in simulations with an error percentage going only upto 3% even for the highest traffic intensity, $\rho = 0.9$ and $n = 50$.

For Erlang-2 task service times (Tables (9)–(14)), the error percentages are within 4% for up to $\rho = 0.8$ and go up to 6% for $n = 20$ and $n = 30$ when $\rho = 0.9$.

When the task service times are according to a Pareto distribution (Tables (15)–(19)), the error percentages are less than 5% for up to $\rho = 0.8$ and less than 10% for $\rho = 0.9$ when $n = 7$ and $n = 10$. However, even when the error percentages are higher, it is encouraging to note that for the Pareto distribution, $\widehat{\widetilde{T}}_n$ lies within the 95% confidence interval of $\widehat{\widetilde{T}}_n^{sim}$. The increase in error percentages can be attributed

to the higher variance of Pareto distribution.

The increase in error percentage with increase in the traffic intensity ρ can be attributed to the higher warm up period requirement to reach steady state for higher traffic intensities. Similarly, increase in the error percentage with increase in n can be attributed to the higher variance of the response time.

The time required for running the simulations ranged from 16 minutes and 45 seconds for $n = 2$, Pareto task service times and $\rho = 0.1$, to 3 hours, 10 minutes and 20 seconds for $n = 50$, exponential task service times and $\rho = 0.9$. Compared to these long simulation run times, the approximation $\widehat{\overline{T}}_n$ can be computed instantaneously for exponential service time distribution while the other distributions require only one simulation to compute the estimate for the other infinitely many values of $\rho \in [0, 1)$.

We now describe approximations for the symmetric n-dimensional fork-join queueing system existing in literature.

2.5.2 Approximation by Nelson and Tantawi

Nelson and Tantawi [47] provided an approximation for the system with exponential service time distribution at the tasks. They observed that a lower bound to the expected response time is given by the maximum of n independent task service times which for the exponential distribution is $\frac{H_n}{\mu}$. Using properties of associated random variables, they showed that an upper bound for the expected response time is given by the expected value of the maximum of the response times of n independent $M/M/1$ queues. This is given by $\frac{H_n}{\mu(1-\rho)}$. The authors used simulations to obtain a scaling approximation between these two bounds using the observation that both the lower and upper bounds increase with n at the same rate, H_n. We denote the approximation obtained using this method by $\widehat{\overline{T}}_n^{NT}$ and this is given by the following expression:

$$\widehat{\overline{T}}_n^{NT} = \left[\frac{H_n}{H_2} + \frac{4}{11}\left(1 - \frac{H_n}{H_2}\right)\rho\right]\frac{12-\rho}{8\mu(1-\rho)}. \tag{8}$$

The estimated mean response time using Equation 8 and the corresponding error percentages are reported in Tables (2)–(8). Based on these, our estimation technique using Conjecture 2 scores over this method in the following ways:

1. The estimation using Equation 8 is valid only for the exponential task service time distribution. Our technique can be applied to other distributions as well.

2. Our method performs better in terms of error percentages in all the instances. In many cases, the difference is quite significant. For example, in the case of $n = 50$ and $\rho = 0.9$, the percentage error of $\widehat{\overline{T}}_{50}$ is -3.008% while that of $\widehat{\overline{T}}_{50}^{NT}$ is 21.015% (Table 8).

3. The error percentages for all the instances for the estimate by Nelson and Tantawi [47] are positive. This leads us to believe that this method gives an upper bound rather than an approximation of the average response time.

2.5.2.1 Approximation by Varma and Makowski

Varma and Makowski [61] considered a system with general inter-arrival and service times. They provided a light traffic approximation using the fact that the individual response times are independent when an arriving job encounters an empty system. For the systems with exponential inter-arrival times and general service times, they provided a conjecture for the heavy traffic limit using diffusion approximation. They extended this conjecture to the case of general inter-arrival and service times. We denote this approximation by $\widehat{\overline{T}}_n^{VM}$. The heavy traffic conjecture is as follows:

$$\lim_{\rho\to 1}\mu(1-\rho)\overline{T}_n(\rho) = \Bigg[H_n + (4V_n - 3H_n - 1)\frac{\sigma_o^2}{\sigma_o^2+\sigma^2} + 2(1+H_n-2V_n)\left(\frac{\sigma_o^2}{\sigma_o^2+\sigma^2}\right)^2\Bigg]\frac{\sigma_o^2+\sigma^2}{2}\mu^2 \quad (9)$$

where σ_o^2 and σ^2 are the variances of the inter-arrival time and service time distributions respectively and $V_n = \sum_{r=1}^{n}\binom{n}{r}(-1)^{r-1}\sum_{m=1}^{r}\binom{r}{m}\frac{(m-1)!}{r^{m+1}}$.

Using the light traffic approximation and the heavy traffic conjecture, they obtained an interpolation approximation for the symmetric fork-join queueing system with general inter-arrival and service times. In our experiments, we compare against this approximation for exponential, gamma and Pareto service time distributions. For exponential service time distribution, $\widehat{\overline{T}}_n^{VM}$ is given by the following:

$$\widehat{\overline{T}}_n^{VM} = \big[H_n + (V_n - H_n)\rho\big]\frac{1}{\mu(1-\rho)}. \quad (10)$$

For gamma service time distribution with mean equal to '1' and shape parameter k equal to '2', $\widehat{\overline{T}}_n^{VM}$ is as follows:

$$\widehat{\overline{T}}_n^{VM} = \left[F_n + \left(\frac{1}{6} - \frac{H_n}{12} + \frac{2}{3}V_n - F_n\right)\rho\right]\frac{1}{\mu(1-\rho)},\ n = 2,3,\ldots \quad (11)$$

where $F_n = \sum_{r=1}^{n}\binom{n}{r}(-1)^{r-1}\sum_{m=0}^{r}\binom{r}{m}\frac{(m)!}{2r^{m+1}}$.

For Pareto service time distribution, $\widehat{\overline{T}}_n^{VM}$ is given by the following expression:

$$\widehat{\overline{T}}_n^{VM} = \frac{M_n + g_1\mu\rho}{\mu(1-\rho)} \quad (12)$$

where $g_1 = \left[H_n + (4V_n - 3H_n - 1)\frac{\sigma_o^2}{\sigma_o^2+\sigma^2} + 2(1+H_n-2V_n)\left(\frac{\sigma_o^2}{\sigma_o^2+\sigma^2}\right)^2\right]\frac{\sigma_o^2+\sigma^2}{2}\mu^2 - M_n$.

Based on results in Tables (2)–(19), we compare our technique against this technique as follows:

1. With exponential and Erlang-2 distributions, our method performs better in terms of the percentage error for higher values of n and ρ. For example, when $n = 50$, $\rho = 0.9$ and service times are exponentially distributed, the error percentage of $\widehat{\overline{T}}_{50}$ is -3.008% while that of $\widehat{\overline{T}}_{50}^{VM}$ is 9.556 (Table 8). For the cases where $\widehat{\overline{T}}_{n}^{VM}$ performs better, which happens for low values of ρ, the error percentages obtained from both $\widehat{\overline{T}}_{n}^{VM}$ and $\widehat{\overline{T}}_{n}$ are less than 1% and perform comparably.

2. This technique performs well only when the service time distributions are exponential-like. When the service times are distributed according to a Pareto distribution, the utility of the approximation using Equation 12 is questionable. The error percentages go up to -44.752% when $n = 10$ and $\rho = 0.8$ (Table 19). This value is very high when compared to the error percentage obtained from $\widehat{\overline{T}}_{n}$ which is 9.643%.

3. The accuracy of the approximation $\widehat{\overline{T}}_{n}^{VM}$ for low values of ρ can be explained by the fact that this is an interpolation approximation and the exact mean response time is known when $\rho = 0^{+}$.

4. Our conjectures contradict the heavy traffic conjecture of Varma and Makowski [61]. The difference in error percentages lends more confidence to Conjectures 1 and 2.

2.5.3 Approximation by Ko and Serfozo

Ko and Serfozo [36] gave an approximation for the expected response time in a fork-join queueing system with exponential inter-arrival and service times, where

each node acts as a queueing station with s servers, $s \geq 1$. They used elementary probability laws and properties of associated random variables to come up with this approximation. Since we are concerned with single server queues, we denote this approximation, for $s = 1$, by $\widehat{\overline{T}}_n^{KS}$ and this is same as the exact result by Nelson and Tantawi [47] for $n = 2$. For $n \geq 3$, $\widehat{\overline{T}}_n^{KS}$ is given by:

$$\widehat{\overline{T}}_n^{KS} = \left[\left(\frac{3}{2} - \frac{\rho}{4}\right) + \left(\frac{5}{4} - \frac{\rho}{8}\right)\left(\frac{5}{4} - \frac{\rho}{4}\right)\left(H_n - \frac{3}{2}\right)\right]\frac{1}{\mu(1-\rho)} \tag{13}$$

Our method scores over this method for the same reasons as enumerated in Section 2.5.2. However, in this case, the error percentages are exorbitantly high even for low values of ρ and n. When $n = 3$ and $\rho = 0.1$ for the exponential service time distribution (Table 2), the error percentage is 10.131% which is the lowest observed. This goes up to 79.818% when $n = 50$ and $\rho = 0.9$ making the utility of the approximation by Ko and Serfozo [36] very highly questionable.

2.5.4 Approximation by Thomasian and Tantawi

Thomasian and Tantawi [57] extended the work of Nelson and Tantawi [47] to come up with an approximation for the expected response time of symmetric fork-join queues with exponential inter-arrival times and general service times. We denote this approximation by $\widehat{\overline{T}}_n^{TT}$ and it is of the form:

$$\widehat{\overline{T}}_n^{TT} = \overline{T}_1 + \frac{M_n - \frac{1}{\mu}}{\sigma_G}\alpha_n(\rho) \tag{14}$$

where, $\overline{T}_1$, μ and M_n are as defined in Sections 2.2 and 2.3 respectively, σ_G is the standard deviation of the service time CDF $G(.)$, and $\alpha_n(\rho)$ is a possibly non-linear function of the the traffic intensity ρ and depends on $G(.)$. Estimation of $\alpha_n(\rho)$ requires many simulations to obtain enough data points to be able to fit a function

on the values of $\alpha_n(\rho)$.

The advantages of our method over than of Thomasian and Tantawi [57] are as follows:

1. The computation of $\widehat{\overline{T}}_n^{TT}$ requires a large number of simulations to fit a curve on $\alpha_n(\rho)$. However, our method requires no simulations for the case of exponential service times and exactly one simulation for other service time distributions. Since simulations take up a lot of time and computing resources, therefore ours is more resource efficient.

2. Our method performs significantly better in terms of the error percentages observed in Tables (9)–(14). For example, when $n = 15$ and $\rho = 0.9$ (Table 12), the error percentage obtained from $\widehat{\overline{T}}_{15}^{TT}$ is -35.127 while that obtained from $\widehat{\overline{T}}_{15}$ is 4.909.

From the discussion presented above,it can be concluded that the superior performance of our method when compared to those existing in literature is indisputable.

2.6 Conclusions

In this chapter, we analyzed the symmetric n-dimensional fork-join queueing system. Even though this system has been around in queueing literature for more than thirty years, its performance analysis has remained a very difficult problem due to dependence between random variables that arise as a result of synchronized arrivals to the queues. We present a conjecture that is strongly supported by simulations and use that to accurately predict the mean response times. We compare this approximation against those in literature and present strong arguments and results in favor of our method. The utility of the results in this chapter lies in making informed decisions on system design. The fork-join queueing system is stable when the arrival

rate of jobs is less than the service rate. However, the response time increases in a continuous manner and converges to infinity as the arrival rate gets close to the service rate. Consequently, even when the system is stable the response times can become inordinately large as the arrival rate increases and gets closer to the value of the service rate. However, given a target mean response time of the system, the results presented here can be used to estimate the maximum allowable arrival rate to the system by simply solving a linear equation. In the next chapter, we extend these methods to other forms of fork-join queueing networks.

3. MORE ON FORK-JOIN QUEUEING NETWORKS

In Chapter 2 we considered the basic form of fork-join queues. This model, though useful as it is in many applications, is restrictive. In this chapter, we relax three aspects of the model in Chapter 2: (i) queueing network structure, (ii) symmetric tasks and (iii) number of tasks that need to be complete before the job exits the system. We analyze fork-join queueing network models constructed by relaxing each of the above constraints individually.

We first provide a brief review of the literature on other fork-join queueing network models.

3.1 Literature Review

The first variation to the symmetric n-dimensional fork-join queueing network was introduced by Baccelli et al [6]. This work analyzed acyclic fork-join queueing networks *i.e.* queueing systems whose precedence requirements result in a queueing network with no cycles. They derived bounds for the response time based on properties of associated random variables and stochastic ordering. However, these bounds are hard to compute in general and do not perform well in terms of distance from the simulated mean response time, as shown by Kemper and Mandjes [34]. Baccelli and Liu [3] computed bounds based on similar concepts for the case of a fork-join queueing system in which multiple tasks are assigned to the same server. This analysis is similar to that of a multi-class queue. Using techniques similar to Baccelli et al [6], Kumar and Shorey [37] derived bounds for fork-join queueing networks in which the number of tasks created at a forking station is a random number which takes a value less than or equal to the number of subsequent task servers available. Nelson et al [48] compared this system with one that has a centralized queue instead

of a separate queue for each task. They modeled the latter system as a $M^X/M/n$ system with synchronization constraints. Their experiments suggest that systems with centralized queues lead to smaller response times as compared to systems with separate queues. Lee and Katz [40] considered a closed fork-join system with variable number of tasks. In this case the number of jobs in the system and the number of tasks in each job are fixed. They gave an approximation technique for the response time and justified their approximations using simulations. Ammar and Gershwin [2] considered a fork-join queueing network with limited buffer capacities and blocking. They showed equivalence relationships between some fork-join networks and more tractable queueing systems such as tandem queues. The results from such queues can then be used for fork-join queues. However, this equivalence relationship is useful for only a very limited set of fork-join queueing networks. Gershwin [27] further extended this work to an acyclic fork-join queueing network with limited buffer space and blocking. In his work an approximate solution was developed to compute the throughput of the system using decomposition of the network into smaller networks. However, this method does not perform well if the buffer sizes are large. Dallery et al [17] showed reversibility properties of fork-join queueing networks with limited buffer capacities and blocking. The throughput of such a network is proved to be a concave function of the buffer sizes and the initial number in each buffer. These properties were consequently used to construct throughput optimal networks. Li and Xu [41] provided approximations and bounds for fork-join queueing systems with finite buffer spaces. However, the performance of these bounds and approximations was not analyzed. Guide Jr. et al [29] considered a heterogeneous and multi-class fork-join queue with only two servers and two classes of jobs. One job consists of two tasks which can be processed in parallel at the two servers and one job requires service at only one server. This work provided an approximation for the

average weighted job response time by approximating the correlated waiting times at the two servers as independent waiting times. Duda and Czachórski [21] derived bounds and approximations for closed fork-join queueing networks. This approximation is based on representing the system as a single server queueing system with state dependent service rate. Varki [59] and Varki [60] also derived approximations and bounds respectively for closed fork-join queueing networks. The approximations were derived using mean-value approach and the upper bound in [60] was derived using basic properties of the underlying Markov chain.

We now present three models of fork-join queues. There have been bounds proposed in literature for each of these systems. However, to the best of our knowledge, no approximations are known for these systems.

In the next section, we relax the restriction on the network structure and present our work on symmetric tandem fork-join queueing networks.

3.2 Symmetric Tandem Fork-Join Queueing Network

In a symmetric tandem fork-join queueing network, jobs arrive according to a Poisson process with rate λ. On arrival, each job splits into n tasks. Each task consists of l sub-tasks in series each of which is served at its own queueing station which consists of a single server operating under FIFO service discipline. Completion of task i corresponds to completion of the last of the l sub-tasks constituting task i, $1 \leq i \leq n$. After completion of all the n tasks, the job is said to be complete and it departs from the system. The service time at each station is exponentially distributed with mean $1/\mu$ and these service times are independent across sub-tasks as well as jobs. Each sub-task is processed by a single server operating under FIFO service discipline. This network model is shown in Figure 7. The symmetric tandem fork-join queueing system falls into a wider class of queueing systems known as the

acyclic fork-join queueing system. Similar to the symmetric n-dimensional fork-join queue, Baccelli et al [6] showed that this system is also stable *iff* $\lambda < \mu$. They derived bounds for the mean response time of the system. However, to the best of our knowledge, no approximations are available for general (non-heavy) traffic intensities for this system.

In this work, we extend the conjectures in Chapter 2 to estimate the mean response time in steady state for this system.

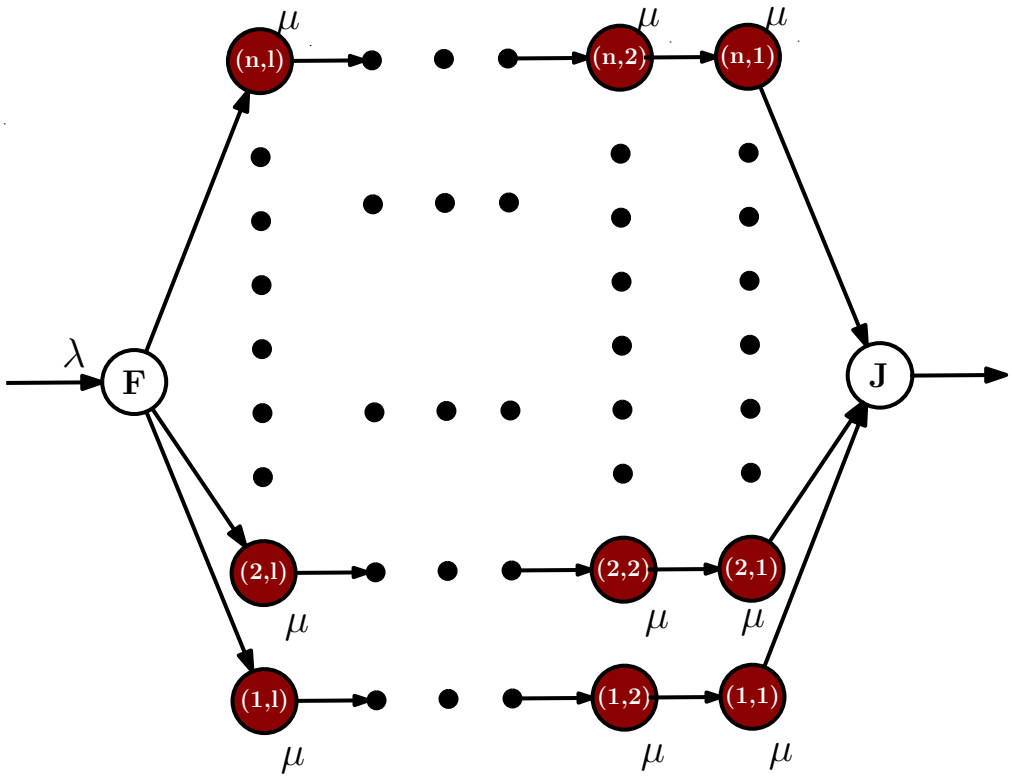

Figure 7: Symmetric Tandem Fork-Join Queueing Network

3.2.1 Response Time Estimation

Following the idea in Chapter 2, we provide a conjecture that can be used to estimate the expected response time in steady state of the symmetric tandem fork-join queueing network.

Conjecture 3. *The mean response time of the symmetric tandem fork-join queueing network is linear with respect to* $\frac{\rho}{1-\rho}$*, i.e. the slope and intercept are independent of the arrival rate,* λ *and the service rate,* μ*.*

$$E[T_{(n,l)}] = \frac{m_{(n,l)}\rho}{\mu(1-\rho)} + \frac{M_{(n,l)}}{\mu} \tag{15}$$

where, $m_{(n,l)}$ *is a parameter independent of* λ *and* $M_{(n,l)}$ *is the mean of the maximum of* n *iid* $Erlang(l,1)$ *random variables.*

Intuition behind Conjecture 3 In the case of symmetric tandem fork-join queues, we denote the number of sub-task queueing stations remaining for the nth task to get completed at the time when the last among the first $n-1$ tasks joins the join buffer by R_l. The value of the expectation of R_l is the expected number of sub-task stations that have been crossed by a job in the slowest task queue when it's first $n-1$ tasks are complete subtracted from the total number of tasks l. This time between the arrival of the job and when the value of R_l is observed depends on the time taken by all the sub-tasks that are complete for that job in the system at the moment when the second from the last task is complete. Since the arrivals are synchronized at all task queues and all the sub-tasks have identical service time distribution, if the arrival intensity changes, it changes for all the subtasks in the same manner. We conjecture that the time interval after which R_l is observed changes appropriately with the change in the arrival rate. Therefore, R_l does not depend

on the arrival rate. In fact we prove that this is true when $n = 2$ in Lemma 1. If this holds true, then the time required at the remaining R_l sub-task queueing stations is similar to the excess time required by the last task to get completed after the completion of the second from the last to finish in the case of $l = 1$. In our experiments in Chapter 2 we observed that this excess time is linear with respect to $\frac{\rho}{1-\rho}$. This leads to Conjecture 3.

We now show that the expectation of R_l above is indeed independent of λ and μ when $n = 2$.

Lemma 1. *In a symmetric tandem fork-join queueing network with number of tasks $n = 2$ and the number of sub-tasks of each task, l, the mean number of sub-tasks of the second task remaining to be completed when the first task is completed, R_l, is independent of the arrival intensity λ and is given by:*

$$E[R_l] = \sum_{s=1}^{l} \frac{s}{2^{2l-s}} \binom{2l-s-1}{l-s} = M_{(2,l)} - l - 1 \tag{16}$$

Proof. We denote the time required to complete the first task in steady state by $T^{(1)}$. Each sub-task in the second task requires a random amount of time that is exponentially distributed with rate $\mu(1-\rho)$. Conditioned on knowing $T^{(1)}$, the number of completed sub-tasks of the second task is a Poisson random variable. The expectation is given by the following:

$$E[R_l] = E\Big[E[R_l|T^{(1)}]\Big] = E\left[\sum_{s=1}^{l} s.\frac{e^{-\mu(1-\rho)T^{(1)}}\{\mu(1-\rho)T^{(1)}\}^{l-s}}{(l-s)!}\right] \tag{17}$$

Since the distribution of $T^{(1)}$ is known to be Erlang $(l, \mu(1-\rho))$, we obtain the following result:

$$E\left[\frac{e^{-\mu(1-\rho)T^{(1)}}\{\mu(1-\rho)T^{(1)}\}^{l-s}}{(l-s)!}\right]$$
$$= \int_0^\infty \frac{e^{-\mu(1-\rho)T^{(1)}}\{\mu(1-\rho)T^{(1)}\}^{l-s}}{(l-s)!} . \frac{\{\mu(1-\rho)\}^{l}\{T^{(1)}\}^{l-1}e^{-\mu(1-\rho)T^{(1)}}}{(l-1)!} dT^{(1)}$$
$$= \frac{1}{2^{2l-s}}\binom{2l-s-1}{l-s}$$

Substituting in Equation (17), we obtain the first part of the result in Lemma 1. Since $E[R_l]$ is independent of ρ, therefore, this value remains the same when $\rho \to 0^+$ which leads to the second part of Lemma 1. □

Remark 2. *Similar to the case of symmetric n-dimensional fork-join queues, our experiments show that the value of the parameter m_n in Equation 15 does not depend on the sub-task service rate μ. Therefore, for a system with fixed number of parallel tasks n and sub-tasks l, the value of m_n remains unchanged with changes in service rate μ and job arrival rate λ. Since these are the system parameters that change frequently, therefore, once $m_{(n,l)}$ is estimated using one simulation of the system, it can be treated as a constant for those values of n and l.*

Remark 3. *Conjecture 3 is satisfied when the number of tasks $n = 1$. In this case, the system is a tandem $M/M/1$ queueing network. The mean response time in steady state is $\frac{l\rho}{\mu(1-\rho)} + \frac{l}{\mu}$, with $m_{(1,l)} = l$ and $M_{(1,l)} = \frac{l}{\mu}$.*

We demonstrate the performance of Conjecture 3 in Figure 8. The linear relationship is obvious from the straight line observed.

Conjecture 3 leads to an approximation technique similar to that in Chapter 2 for the mean response time in steady state of the symmetric tandem fork-join queueing system. For completion, we present this is Algorithm 2. For brevity, we denote

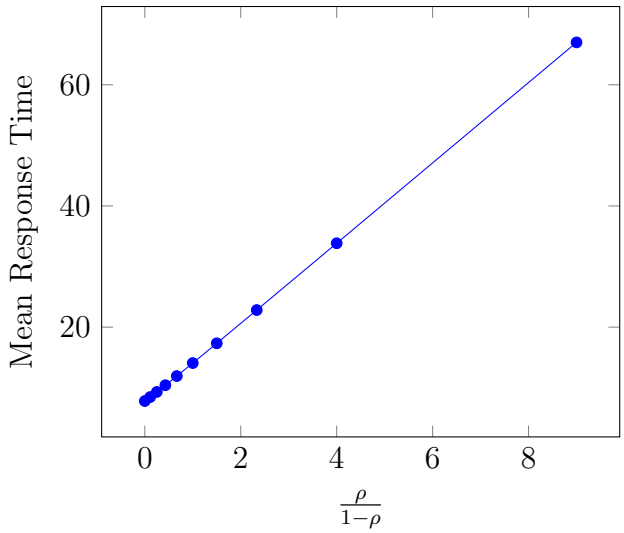

Figure 8: Plot of simulated average response time vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $l = 5$ and $\mu = 1$

$E[T_{(n,l)}]$ by $\overline{T}_{(n,l)}$, our estimate by $\widehat{\overline{T}}_{(n,l)}$ and the estimate obtained by simulations by $\widehat{\overline{T}}^{sim}_{(n,l)}$. We also denote the estimate of the parameters $m_{(n,l)}$ in Equation (15) by $\widehat{m}_{(n,l)}$.

Algorithm 2 Computation of $\widehat{\overline{T}}_{(n,l)}$

Input: Number of parallel tasks, n; Number of sub-tasks, l

Output: Expression for $\widehat{\overline{T}}_{(n,l)}$ $\forall \rho \in [0, 1)$, *i.e.* $\forall \lambda \in [0, \mu)$, $\mu > 0$.

1: Simulate the system for $\rho = 0.5$ and $\mu = 1$. Estimate expected response time $\widehat{\overline{T}}^{sim}_{(n,l)}(0.5)$.

2: Compute $M_{(n,l)}$

3: Solve linear equation for $\widehat{m}_{(n,l)}$: $\widehat{\overline{T}}^{sim}_{(n,l)}(0.5) = \frac{0.5\widehat{m}_{(n,l)}}{\mu(1-0.5)} + M_{(n,l)}$

4: **return** $\widehat{\overline{T}}_{(n,l)}(\rho) = \frac{\rho\widehat{m}_{(n,l)}}{\mu(1-\rho)} + M_{(n,l)}$

We now demonstrate the use of Algorithm 2 with a numerical example.

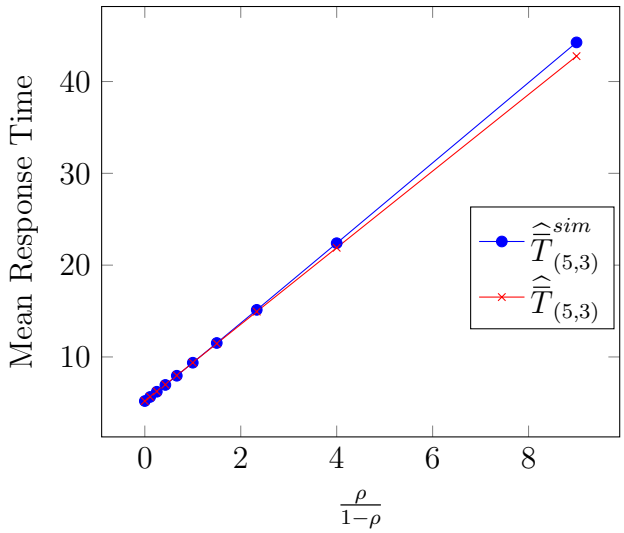

Figure 9: Plot of simulated and predicted average response times vs. $\frac{\rho}{1-\rho}$ for $n = 5$, $l = 5$ and $\mu = 1$

3.2.2 Numerical Example

Consider a system with $n = 5$ and $l = 3$. When we run a simulation for this system with $\rho = 0.5$ and $\mu = 1$, we get the value of $\widehat{\overline{T}}^{sim}_{(3,5)}(0.5)$ to be equal to 9.37 time units. Computing the expectation of the maximum of five independent Erlang-$(3, 1)$ random variables, we get $M_{(5,3)} = 5.19$. We solve the following equation for $\widehat{m}_{(5,3)}$: $9.37 = \frac{0.5\widehat{m}_{(5,3)}}{1(1-0.5)} + 5.19$, to obtain $\widehat{m}_{(5,3)} = 4.17$. Therefore, $\forall \rho \in [0, 1)$ and $\mu \geq 0$, we obtain the following approximation for $\overline{T}_{(3,5)}$: $\widehat{\overline{T}}_{(3,5)} = \frac{4.17\rho}{\mu(1-\rho)} + \frac{5.19}{\mu}$. We demonstrate the performance of Algorithm 2 in Figure 9.

3.2.3 Experimental Results

In this section we present results to compare the performance of Algorithm 2 with the estimates of mean response times obtained from simulations. Results are reported for $l = 2, 3, 4$ and 5 and $n = 2$, 5 and 10 (Tables (21)–(26)). A sub-task service rate of $\mu = 1$ was assumed. In the tables in this section, for the values of n and

l for which results are reported, $\widehat{\overline{T}}^{sim}_{(n,l)}$ denotes the expected response time estimate obtained using simulations, $\widehat{\overline{T}}_{(n,l)}$ denotes that obtained using Algorithm 2 and % Error denotes the error percentage observed. The calculation of this value is similar to that in Equation (7). Results are reported for $\rho = 0.1$ to 0.9 with increments of 0.1 excluding $\rho = 0.5$. This has been excluded since this simulated mean response time estimate for $\rho = 0.5$ was used in the estimation of the response times for other values of ρ using Algorithm 2. The set up of simulations is similar to that in Chapter 2. For each simulation result, 95% confidence intervals are reported.

When $n = 2$ (Tables (21)–(22)), the error percentages are within 2%, the highest being 1.912% when $l = 5$ and $\rho = 0.9$. When $n = 5$ (Tables (23)–(24)), the error percentages fall within 5% with the maximum being 4.117% when $l = 5$ and $\rho = 0.9$. Lastly, when $n = 10$ (Tables (25)–(26)), the error percentages are within 6%, the maximum being 5.617% when $l = 5$ and $\rho = 0.9$. The increase in error percentages with increase in ρ can again be attributed to the fact that more time is required to approximate steady state accurately in that case.

The time required for simulations ranged from 37 minutes and 29 seconds when $n = 2$, $l = 2$ and $\rho = 0.1$ to 4 hours, 33 minutes and 45 seconds when $n = 10$, $l = 5$ and $\rho = 0.9$. These huge simulation times highlight the importance of our work in approximating the average response time in less than a second once the value of $m_{(n,l)}$ is known for a given number of tasks and sub-tasks. The importance of our contribution increases with the fact that to the best of our knowledge, there are no approximation methods available for estimating the steady state average response time of the symmetric tandem fork-join queueing system, at general traffic intensities, available in literature.

Table 21: Comparison with simulation results for $n = 2$, $l = 2$ and $n = 2$, $l = 3$

ρ	$\widehat{\overline{T}}^{sim}_{(2,2)}$	$\widehat{\overline{T}}_{(2,2)}$	% Error	$\widehat{\overline{T}}^{sim}_{(2,3)}$	$\widehat{\overline{T}}_{(2,3)}$	% Error
0.1	3.02±0.000	3.03	0.077	4.32±0.000	4.33	0.133
0.2	3.37±0.000	3.37	0.112	4.81±0.001	4.82	0.202
0.3	3.81±0.001	3.81	0.115	5.44±0.001	5.45	0.203
0.4	4.40±0.001	4.40	0.080	6.29±0.001	6.30	0.137
0.6	6.48±0.003	6.47	-0.122	9.26±0.003	9.24	-0.212
0.7	8.56±0.005	8.54	-0.284	12.25±0.006	12.19	-0.496
0.8	12.74±0.013	12.68	-0.484	18.25±0.013	18.09	-0.865
0.9	25.26±0.057	25.09	-0.682	36.24±0.049	35.78	-1.284

Table 22: Comparison with simulation results for $n = 2$, $l = 4$ and $n = 2$, $l = 5$

ρ	$\widehat{\overline{T}}^{sim}_{(2,4)}$	$\widehat{\overline{T}}_{(2,4)}$	% Error	$\widehat{\overline{T}}^{sim}_{(2,5)}$	$\widehat{\overline{T}}_{(2,5)}$	% Error
0.1	5.59±0.000	5.60	0.183	6.84±0.001	6.85	0.226
0.2	6.22±0.001	6.24	0.276	7.60±0.001	7.63	0.335
0.3	7.03±0.001	7.05	0.278	8.60±0.001	8.63	0.333
0.4	8.13±0.002	8.14	0.189	9.93±0.001	9.96	0.220
0.6	11.99±0.005	11.95	-0.281	14.66±0.004	14.62	-0.331
0.7	15.87±0.010	15.76	-0.662	19.42±0.007	19.27	-0.764
0.8	23.65±0.025	23.38	-1.137	28.96±0.022	28.59	-1.289
0.9	47.03±0.096	46.25	-1.662	57.64±0.078	56.54	-1.912

Table 23: Comparison with simulation results for $n = 5$, $l = 2$ and $n = 5$, $l = 3$

ρ	$\widehat{\overline{T}}^{sim}_{(5,2)}$	$\widehat{\overline{T}}_{(5,2)}$	% Error	$\widehat{\overline{T}}^{sim}_{(5,3)}$	$\widehat{\overline{T}}_{(5,3)}$	% Error
0.1	4.14±0.000	4.15	0.211	5.64±0.000	5.66	0.313
0.2	4.56±0.000	4.58	0.327	6.21±0.000	6.24	0.483
0.3	5.11±0.001	5.12	0.339	6.95±0.000	6.99	0.498
0.4	5.84±0.001	5.86	0.234	7.95±0.001	7.98	0.341
0.6	8.45±0.002	8.42	-0.382	11.52±0.002	11.46	-0.535
0.7	11.08±0.005	10.98	-0.918	15.13±0.006	14.94	-1.276
0.8	16.36±0.014	16.09	-1.608	22.39±0.014	21.89	-2.218
0.9	32.25±0.048	31.45	-2.483	44.26±0.064	42.77	-3.383

Table 24: Comparison with simulation results for $n = 5$, $l = 4$ and $n = 5$, $l = 5$

ρ	$\widehat{\overline{T}}^{sim}_{(5,4)}$	$\widehat{\overline{T}}_{(5,4)}$	% Error	$\widehat{\overline{T}}^{sim}_{(5,5)}$	$\widehat{\overline{T}}_{(5,5)}$	% Error
0.1	7.07±0.000	7.10	0.395	8.46±0.001	8.50	0.465
0.2	7.78±0.000	7.83	0.610	9.30±0.001	9.37	0.715
0.3	8.71±0.001	8.76	0.627	10.41±0.001	10.49	0.728
0.4	9.97±0.002	10.01	0.426	11.92±0.001	11.98	0.490
0.6	14.46±0.004	14.37	-0.652	17.33±0.003	17.21	-0.732
0.7	19.02±0.007	18.73	-1.521	22.82±0.006	22.43	-1.689
0.8	28.18±0.018	27.45	-2.597	33.84±0.021	32.88	-2.838
0.9	55.76±0.051	53.62	-3.841	66.99±0.099	64.24	-4.117

Table 25: Comparison with simulation results for $n = 10$, $l = 2$ and $n = 10$, $l = 3$

ρ	$\widehat{\overline{T}}^{sim}_{(10,2)}$	$\widehat{\overline{T}}_{(10,2)}$	% Error	$\widehat{\overline{T}}^{sim}_{(10,3)}$	$\widehat{\overline{T}}_{(10,3)}$	% Error
0.1	4.99±0.000	5.01	0.314	6.62±0.000	6.65	0.432
0.2	5.47±0.000	5.49	0.493	7.24±0.001	7.29	0.676
0.3	6.08±0.001	6.12	0.518	8.05±0.001	8.11	0.702
0.4	6.92±0.001	6.94	0.361	9.16±0.001	9.21	0.484
0.6	9.90±0.003	9.85	-0.584	13.14±0.004	13.04	-0.769
0.7	12.93±0.005	12.75	-1.413	17.19±0.007	16.88	-1.826
0.8	19.02±0.015	18.55	-2.492	25.35±0.016	24.55	-3.153
0.9	37.39±0.056	35.96	-3.833	49.89±0.084	47.57	-4.659

Table 26: Comparison with simulation results for $n = 10$, $l = 4$ and $n = 10$, $l = 5$

ρ	$\widehat{\overline{T}}^{sim}_{(10,4)}$	$\widehat{\overline{T}}_{(10,4)}$	% Error	$\widehat{\overline{T}}^{sim}_{(10,5)}$	$\widehat{\overline{T}}_{(10,5)}$	% Error
0.1	8.15±0.000	8.19	0.532	9.62±0.001	9.68	0.614
0.2	8.91±0.001	8.98	0.827	10.52±0.001	10.62	0.950
0.3	9.91±0.001	10.00	0.856	11.70±0.001	11.82	0.972
0.4	11.28±0.002	11.35	0.586	13.33±0.002	13.42	0.657
0.6	16.23±0.005	16.08	-0.905	19.21±0.004	19.02	-0.994
0.7	21.26±0.009	20.81	-2.103	25.20±0.009	24.63	-2.292
0.8	31.39±0.017	30.28	-3.569	37.27±0.022	35.83	-3.857
0.9	61.94±0.079	58.67	-5.273	73.59±0.118	69.46	-5.617

3.3 Heterogeneous Fork-Join Queueing Systems

In Chapter 2, we provided an approximation for the expected response time in steady state of the symmetric n-dimensional fork-join queueing system. The restriction of a symmetric system requires all the service time random variables to follow identical distributions. In this section we relax this requirement and allow the task service times to take different mean values. We now define the problem formally.

3.3.1 Problem Description

Similar to the systems that we have seen so far, jobs arrive into the system following a Poisson process with rate λ. Each job is split instantaneously into n tasks. The tasks are processed at n single server queueing stations operating under FIFO service discipline. The service times of the tasks are independent across jobs and across tasks belonging to the same job. However, these tasks are not identically distributed. Task i, $i = 1, \ldots, n$ is exponentially distributed with rate μ_i. A job is considered complete and leaves the system when all the n tasks are complete. It follows from the work of Baccelli et al [6] that this system is stable *iff* $\lambda < \min_{i=1,\ldots,n} \mu_i$. To our knowledge, there are no approximations available in literature for the steady state response time of this system when $n > 2$. We now present an estimation technique for the steady state mean response time of this system.

3.3.2 Response Time Estimation

We provide a conjecture which, like in systems previously described, can be used to estimate the mean response time in steady state. In this system, each task response time is exponential with rate $\mu_i - \lambda$. For a given arrival intensity λ and values of $i = 1, \ldots, n$, we denote the expectation of the maximum of i *independent* exponential random variables with rate $\mu_i - \lambda$ by $M_i(\lambda)$. We denote the steady state response

time random variable of this system by H_n.

Conjecture 4. *The mean response time in steady state of the heterogeneous fork-join queueing network, $E[S_n]$ is given by:*

$$E[S_n] = \sum_{i=1}^{n} (M_i(\lambda) - M_{i-1}(\lambda))(\mu_i - \lambda) \left[\frac{\lambda m_n^{(i)}}{\mu_i(\mu_i - \lambda)} + \frac{1}{\mu_i} \right] \tag{18}$$

where, $m_n^{(i)}$ are parameters independent of λ, $m_n^{(1)} = 1$ and $M_0(\lambda) = 0$, $\forall \lambda \in [0, \min_{i=1,\ldots,n} \mu_i)$.

Remark 4. *As an example, for $n = 2$, $(M_2(\lambda) - M_1(\lambda))(\mu_2 - \lambda) = \frac{\mu_1 - \lambda}{\mu_1 + \mu_2 - 2\lambda}$. Therefore, we have*

$$E[S_2] = \frac{1}{\mu_1 - \lambda} + \frac{\mu_1 - \lambda}{\mu_1 + \mu_2 - 2\lambda} \left[\frac{\lambda m_2^{(2)}}{\mu_2(\mu_2 - \lambda)} + \frac{1}{\mu_2} \right]. \tag{19}$$

If we substitute $\mu_1 = \mu_2 = \mu$ in Equation (19), we obtain the expression for the symmetric 2-dimensional fork-join queue given in Nelson and Tantawi [47], where $m_2^{(2)} = \frac{3}{4}$. On the other hand, if $\frac{1}{\mu_1 - \lambda} >> \frac{1}{\mu_2 - \lambda}$, $E[S_2] \approx \frac{1}{\mu_1 - \lambda}$. This is true due to the following reason: if one service rate is much higher than the other, the response time will be dominated by the contribution of the slower queue, especially as the difference in the response times in the two queues increases with increase in the traffic intensity.

Intuition behind Conjecture 4. Consider the symmetric fork-join queue of Chapter 2 with $n = 2$ in steady state. Once the task with index 1 of any job is completed, if it is the last of the two tasks to get completed, it departs from the system. If not it waits in a join buffer for the other task to get completed. If we denote the excess time that it waits for by R, it was observed that:

$$E[R] = \frac{3\lambda}{8\mu(\mu - \lambda)} + \frac{1}{2\mu}. \tag{20}$$

The mean synchronization time consists of two parts. One is the mean service time of the second task. Due to the memoryless property of the exponential distribution, the expectation of this is equal to $\frac{M_2(0)-M_1(0)}{\mu_2}$. The other is the workload in front of the second task of the job in the second task queue when the first finishes. It was shown by Nelson and Tantawi [47] for $n = 2$ and observed in Chapter 2 for $n > 2$ that this residual expected workload is directly proportional to the average steady state workload in the second queue, *i.e.* $\frac{\lambda}{\mu(\mu-\lambda)}$ and the proportionality constant is independent of λ. We extend this to the heterogeneous case to obtain Conjecture 4.

We now use Conjecture 4 to come up with an algorithm to estimate the response time in steady state of the heterogeneous fork-join system. In this case, we have $n-1$ parameters that need to be estimated. Therefore, we require $n-1$ values of the arrival intensity λ for which we simulate the system. Using these values, we solve $n-1$ linear equations to obtain the values of these parameters. These can then be used to estimate the mean response time in steady state for all other arrival intensities. We present this algorithm formally in Algorithm 3. For brevity, we denote $E[S_n]$ by $\overline{S}_n$, our estimate by $\widehat{\overline{S}}_n$ and the estimate obtained by simulations by $\widehat{\overline{S}}_n^{sim}$. We also denote the estimate of the parameters m_n in Equation (15) by $\widehat{m}_n$.

Remark 5. *Suppose we have a network with n tasks, each with service rate μ_i, $i = 1, \ldots, n$. If the network controlling authority is considering adding a new task with service rate μ_{n+1} while keeping the initial network the same, the structure of Algorithm 3 is such that the estimated values of $m_n^{(i)}$ will remain the same in the new network with $n+1$ tasks. Only one simulation will be needed to estimate $m_{n+1}^{(n+1)}$.*

3.3.3 Numerical Example and Results

We now demonstrate the use of Algorithm 3 with a numerical example. We consider the case of $n = 2$, $\mu_1 = 1$ and $\mu_2 = 1.5$. Conjecture 4 for this case takes

Algorithm 3 Computation of $\widehat{\widehat{S}}_n$

Input: Number of parallel tasks, n; Arrival rate λ, Service rates μ_i, $i = 1, \ldots, n$

Output: Expression for $\widehat{\widehat{S}}_n$ $\forall \lambda \in [0, \min_{i=1,\ldots,n} \mu_i)$

1: Choose any $n-1$ values of $\lambda \in [0, \min_{i=1,\ldots,n} \mu_i)$, $\hat{\lambda}_j$, $j = 2, \ldots, n$. Simulate and estimate expected response times $\widehat{\widehat{S}}_n^{sim}(\hat{\lambda}_j)$.

2: Compute $M_i(\hat{\lambda}_j)$, $\forall i = 1, \ldots, n$ and $j = 2, \ldots, n$.

3: Solve system of linear equations for $\widehat{m}_n^{(i)}$, $i = 1, \ldots, n$: $\widehat{m}_n^{(1)} = 1$; $\widehat{\widehat{S}}_n^{sim}(\hat{\lambda}_j) = \sum_{i=1}^{n} (M_i(\hat{\lambda}_j) - M_{i-1}(\hat{\lambda}_j))(\mu_i - \hat{\lambda}_j) \left[\frac{\hat{\lambda}_j \widehat{m}_n^{(i)}}{\mu_i(\mu_i - \hat{\lambda_j})} + \frac{1}{\mu_i} \right]$, $\forall j = 2, \ldots, n$.

4: **return** $\widehat{\widehat{S}}_n(\lambda) = \sum_{i=1}^{n} (M_i(\lambda) - M_{i-1}(\lambda))(\mu_i - \lambda) \left[\frac{\lambda \widehat{m}_n^{(i)}}{\mu_i(\mu_i - \lambda)} + \frac{1}{\mu_i} \right]$

the form in Equation (19). We choose $\hat{\lambda}_2 = 0.5$ and run simulations to obtain $\widehat{\widehat{S}}_2^{sim}(0.5) = 2.216$ time units. $M_1(0.5) = \frac{1}{1-0.5} = 2$ and $M_2(0.5) = 2.333$. We solve the resulting linear equation to get $\widehat{m}_2^{(2)} = 0.322$. Finally, we substitute this in Equation (19) to obtain the approximations for the steady state mean response time.

We exhibit the performance of Algorithm 3 by comparing the predicted values of the expected response time in steady state against those obtained from simulations. In this section, we report values for various values of the arrival intensity λ as opposed to the traffic intensity ρ like in previous sections. Let $\widehat{\widehat{S}}_n^{sim}$ denote the values of the expected steady state response times obtained using simulations. These are reported along with their 95% confidence interval. Let $\widehat{\widehat{S}}_n$ denote the values obtained using Algorithm 3. Lastly, the error percentages are calculated using Equation (7). Since more than one simulation is required in the running of Algorithm 3, therefore, we present results for more values of the arrival intensity than in previous sections. Traffic intensities are calculated based on the task with the lowest service rate. Results are reported for arrival intensities corresponding to $\rho = 0.05, \ldots, 0.9$ at intervals

of 0.05.

In the first system (Table 27), the parameters are same as that in the numerical example. Only one simulation is required in this case. Simulation results for $\lambda = 0.5$ were used to estimate the unknown parameter. The error percentages are very encouraging with the maximum being 0.454% when the arrival intensity is 0.9.

In the second system (Table 28), $n = 3$, $\mu_1 = 0.5$, $\mu_2 = 1.0$ and $\mu_3 = 1.5$. In this case, it would have been possible to use the values of unknown parameters obtained in the previous system as suggested in Remark 5, and obtain the required predicted response time values by running only one simulation. However, the results reported here are calculated by following Algorithm 3 explicitly. Simulations for arrival intensities of 0.150 and 0.350 were used to estimate the two unknown parameters. In this case too, the error percentages are less than 1% with the maximum being 0.802% when the arrival intensity is 0.450.

We now present a brief preliminary analysis for another fork-join system with promising applications.

3.4 (n,k) Fork-Join Queues

In analyzing the (n, k) fork-join queueing system, we relax the restriction on the number of tasks that need to be processed before the job is considered to be complete. Joshi et al [32] introduced these (n, k) fork-join queues which are encountered in network coding algorithms. Arrivals into the system are according to a Poisson process with rate λ. As in previous sections, the job is split into n tasks that are processed in parallel at n queueing stations operating on a FIFO service discipline. Each task requires independent service times distributed exponentially with rate μ. So far, the system description is identical to that in Chapter 2. However, out of the n tasks, only k need to be finished for the job to be complete. When an entity

Table 27: Comparison with simulation results for $n = 2$, $\mu_1 = 1.0$ and $\mu_2 = 1.5$

λ	$\widehat{\widehat{H}}_2^{sim}$	$\widehat{\widehat{H}}_2$	% Error
0.05	1.32±0.000	1.32	-0.049
0.10	1.38±0.000	1.38	-0.094
0.15	1.45±0.000	1.45	-0.128
0.20	1.52±0.000	1.52	-0.151
0.25	1.61±0.000	1.61	-0.160
0.30	1.71±0.000	1.70	-0.157
0.35	1.82±0.000	1.81	-0.139
0.40	1.94±0.000	1.94	-0.107
0.45	2.10±0.001	2.09	-0.059
0.55	2.49±0.001	2.50	0.072
0.60	2.77±0.001	2.77	0.149
0.65	3.12±0.001	3.12	0.230
0.70	3.58±0.002	3.59	0.312
0.75	4.23±0.003	4.25	0.381
0.80	5.21±0.006	5.23	0.427
0.85	6.84±0.012	6.87	0.454
0.90	10.12±0.031	10.16	0.420

Table 28: Comparison with simulation results for $n = 3$, $\mu_1 = 0.5$, $\mu_2 = 1.0$ and $\mu_3 = 1.5$

λ	$\widehat{\widehat{H}}_3^{sim}$	$\widehat{\widehat{H}}_3$	% Error
0.025	2.53±0.000	2.53	0.013
0.050	2.64±0.000	2.65	0.016
0.075	2.77±0.001	2.77	0.016
0.100	2.91±0.001	2.91	0.012
0.125	3.07±0.001	3.07	0.006
0.175	3.46±0.001	3.46	-0.009
0.200	3.70±0.001	3.70	-0.019
0.225	3.99±0.002	3.99	-0.029
0.250	4.34±0.002	4.34	-0.036
0.275	4.77±0.002	4.77	-0.042
0.300	5.30±0.003	5.30	-0.044
0.325	6.00±0.004	5.99	-0.033
0.375	8.23±0.007	8.23	0.067
0.400	10.19±0.012	10.22	0.198
0.425	13.49±0.021	13.55	0.438
0.450	20.11±0.049	20.27	0.802

reaches the server for service for task i, $1 \leq i \leq n$, if k tasks belonging to that job are already complete, the service time of task i is zero. However, if k tasks are not complete, the entity enters service and remains in service until its service completion or the completion of k tasks of that entity, whichever happens first.

Joshi et al [32] showed that the (n, k) fork-join queueing system is stable *iff* $\lambda < \frac{n\mu}{k}$. They computed bounds for the response time of the system. To the best of our knowledge, this is the only work on this topic.

We denote the workload at a single task station by $T^{(i)}_{(n,k)}$, $1 \leq i \leq n$. Due to symmetry, we have $E[T^{(1)}_{(n,k)}] = E[T^{(2)}_{(n,k)}] = \ldots = E[T^{(n)}_{(n,k)}]$. We denote the steady state response time random variable by $T_{(n,k)}$. We present a conjecture on the relationship between $E[T_{(n,k)}]$ and $E[T^{(1)}_{(n,k)}]$.

Conjecture 5. *The mean response time of the (n, k) fork-join queueing network is linear with respect to the expected steady state response time of a single task.*

$$E[T_{(n,k)}] = m_{(n,k)} \left(E[T^{(1)}] - \frac{k}{n\mu} \right) + \frac{H_n - H_{k-2}}{\mu} \tag{21}$$

where, $m_{(n,k)}$ is a parameter independent of ρ and H_n is the nth harmonic sum.

The term $\frac{H_n - H_{k-2}}{\mu}$ in Equation (21) is the mean response time of an entity that enters an empty system, *i.e.* the expected value of the k-th order statistic of n *i.i.d.* exponential random variables with rate μ. Since we write Equation (21) in terms of the expected response time of a single queue and not the expected workload in a single queue, we subtract the average service time from the expected response time.

Remark 6. *In the previous sections, we encountered systems where the realizations of the service times at any task were independent of those at other tasks. In this system this is not the case since smaller service times at the first k tasks to complete*

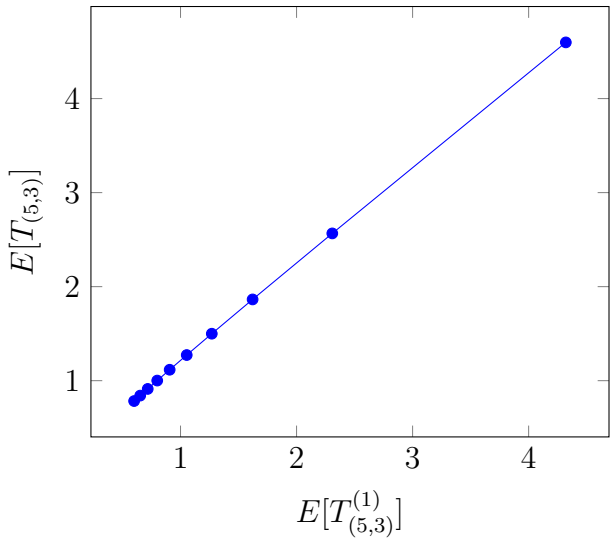

Figure 10: Plot of $E[T^{(1)}_{(5,3)}]$ vs. $E[T_{(5,3)}]$

will result in smaller service times at the remaining $n-k$ tasks too. This opens up the possibility that Conjecture 1 could be extended to systems even when the task service times are correlated.

3.4.1 Comparison with Simulations

We now present preliminary simulation results that support Conjecture 5. In particular, in Figure 10 we present the plot of the simulated average response time of the system on the y-axis and the average time spent by a job at only one task, which we denote by $E[T^{(1)}_{(n,k)}]$ on the x-axis. The straight line supports Conjecture 5.

Unfortunately, unlike the previous sections, we do not have an expression for $E[T^{(1)}_{(n,k)}]$ in Equation (21). This is because the individual queues do not behave as independent systems. For any task queue, the service time of an entity might decrease depending on the service times of other entities in the same as well as other task queues. Since the dependence structure of this system is extremely complicated, estimation of $E[T^{(1)}_{(n,k)}]$ is very difficult. However, efficient bounds on the individual

task response times can be used to obtain good bounds on the total response time of the system and this forms part of our future work.

3.4.2 Conclusions

In this chapter, we provide simple and easy to obtain expressions for fork-join queueing systems that are so hard to analyze that no approximations for the mean response time of these systems exist in literature. We provide conjectures that are strongly suggested to be true in experiments. We use these conjectures to design efficient approximation algorithms for the fork-join queueing networks. This analysis is useful in making decisions on system design.

Algorithm 2 can be used to make decisions on construction of the symmetric tandem fork-join queueing network. For example, division of one task into sub-tasks and having a separate queueing station for each sub-task results in an improvement in the response time. However, if there is a cost (labor cost, for example) for operating each queueing station, then the trade-off between the benefit gained from lower sojourn times and increased cost of keeping the system in operation is not easy to analyze without knowing the increase or decrease in response times with n and l. Prior to this work, the only way to quantify this trade-off was simulations which require huge computing resources and time. Conjecture 3 can now be used to quantify the trade-off between the reduction in response time by increasing the number of subtasks and the cost of operating a queueing station for each sub-task. This knowledge is important for the construction of efficient systems.

Similarly, in the case heterogeneous fork-join queues, we can use Algorithm 3 to estimate the change in response times with addition to or subtraction from the number of parallel tasks by conducting just one extra simulation. This would save computing resources and the time required for running a simulation for each change

in the arrival rate of the new system.

Finally, in the (n, k) fork-join queueing system, if the server starts processing a task and k other tasks get completed before the service completion of that task, then the server has wasted some amount of time and resources in serving a task that left the system before completion. Therefore, it is not obvious that the response time is minimized by sending the entity to all n task processors and waiting for k of them to finish. For example, simulations show that for lower traffic intensities, when $(n, k) = (5, 3)$, the system performs better in terms of lower expected response times than when $(n, k) = (10, 6)$. However, for higher traffic intensities, the system with $(n, k) = (10, 6)$ performs better for the same value of the service rate μ. Therefore, bounds obtained using Conjecture 5 can be used to construct an efficient system based on the requirement.

4. THE MAXIMUM RATIO CLIQUE PROBLEM*

We now move on to the second part of this dissertation. The problem considered in this chapter is encountered when a performance measure is expressed as a ratio of functions of the decision variables and input parameters of the system. We formulate a new fractional programming problem and present solution methodologies.

4.1 Introduction

Given a simple undirected graph $G = (V, E)$ with the set $V = \{1, \ldots, n\}$ of vertices, a *clique* is a subset of vertices inducing a complete subgraph. A *maximal* clique is a clique that is not a subset of a larger clique, and a *maximum* clique is a clique with the maximum possible number of vertices in the graph. Given a non-negative weight w_i associated with each vertex $i \in V$, the *maximum weight clique* problem is to find a clique that maximizes the sum of its vertex weights. The case where $w_i = 1 \; \forall i \in V$ corresponds to the classical maximum clique problem.

The clique concept was originally introduced by Luce and Perry [43] to describe a group of friends, or *cohesive subgroup*, in a social network. It should be noted that maximality by inclusion was required as a part of the original definition, i.e., only maximal cliques were treated as cohesive subgroups. Since then, cliques have found numerous applications in diverse areas, including social network analysis, computational biology and coding theory among others [10]. From an applied perspective, cliques are typically used to model *clusters* in a network representation of a certain complex system, where one wants to maximize the overall weight of a cluster. However, in some cases a fractional objective function provides a more appropriate

*Reprinted with permission from The Maximum Ratio Clique Problem by S. Sethuraman and S. Butenko, 2015. Computational Management Science, 12(1):197-218, Copyright [2015].

description of the underlying goal. For example, in social network analysis, one may be interested in finding cohesive subgroups with the maximum (minimum) average value of a measure of interest, e.g., a group with the highest or lowest average income. As another example, consider a variation of the *market graph* [9], where the vertices represent stocks and the edges correspond to pairs of stocks with negative correlations of price fluctuations. Then a maximal clique describes a diversified portfolio of stocks, which we will refer to as a *clique portfolio*. Assume that the buying price of a fixed number of shares of stock i at time 0 is given by b_i, and the selling price of the same number of shares of the same stock at time 1 is projected to be s_i. Then finding a clique portfolio with the maximum projected return requires maximizing the ratio $(\sum_{i \in C} s_i)/(\sum_{i \in C} b_i)$, where C is a maximal clique in the market graph.

In addition, a fractional objective arises in network-based location models, such as establishment of wind farms. To minimize the effects of wind speed variability, it is required that wind farms be located at places that have a negative correlation with each other in terms of wind speed over time. However, the costs of setting up wind farms might vary widely depending on factors like the size of the farm, the height of wind turbines, and geographical factors. High wind speeds are required to get the maximum possible power output in a wind farm. The problem of finding appropriate locations for wind farms that maximize the overall energy output per dollar invested can be modeled as the problem of finding a maximal clique, where the vertices of the graph represent different possible wind farm configurations for the prospective locations of interest. Then a maximum ratio clique has the highest value of the ratio of the net wind speed to the net cost.

Next, we formally define the problem studied in this chapter.

Definition 1. *Given a simple undirected graph $G = (V, E)$, where each vertex $i \in V$*

is assigned two non-negative rational weights, a_i and b_i, the maximum ratio clique problem (MRCP) is to find a maximal clique C in G that maximizes the quantity $\frac{\sum_{i \in C} a_i}{\sum_{i \in C} b_i}$.

When $b_i = 1$ for all $i \in V$, we obtain a special case of MRCP, which we will refer to as the *maximum average weight clique problem* (MAWCP).

To the best of our knowledge, the maximum ratio clique problem has not been considered in literature. However, fractional objective functions have been studied for many classical problems in combinatorial optimization. Prokopyev et al. [51] proved that the unconstrained fractional 0-1 programming problem is NP-hard. The problem of finding a minimal cost to time ratio cycle was discussed by Dantzig et al. [18] and Lawler [39]. Fox [25] proposed an out-of-killer algorithm for this problem. A special case of this problem, the minimum cycle mean problem, was studied by Karp [33], who gave a polynomial time algorithm. Orlin and Ahuja [50] developed an approximate binary search procedure, which uses a scaling for an assignment problem in its iterations. Dasdan and Gupta [19] discussed polynomial time algorithms to solve this problem for system-performance analysis. Chandrasekaran [12] introduced the minimal ratio spanning tree problem and proposed a polynomial time solution algorithm. Shigeno et al. [54] developed algorithms for the fractional assignment problem and Billionnet [8] developed approximations for the fractional knapsack problem.

Methods for solving general fractional 0-1 programming problems have received considerable attention in the literature. Isbell and Marlow [31] apply Newton's method for solving linear fractional programming problems over a polyhedron. This approach was generalized to special non-linear cases by Dinkelbach [20]. Lawler [39] gave a binary search procedure for solving linear and some cases of nonlinear frac-

tional programming problems. A comparison of these methods along with some modifications was provided by Ibaraki [30]. Radzik [52] gave a modification of Newton's method that has the number of iterations bounded by a polynomial function of the number of variables. Megiddo [44] established the relation between the time complexity of the linear objective version of a combinatorial optimization problem and that of the corresponding fractional objective version. It was shown that if the linear version had an efficient algorithm, then an efficient algorithm could be found for the fractional version. Megiddo's method is useful for the minimum ratio cycle, minimum ratio spanning tree, minimum ratio (simple) path and maximum ratio weighted matching problems because it gives an algorithm running in polynomial time in the number of vertices in the graph. Wu [64] proposed a method for solving a linear fractional 0-1 programming problem by converting it into a mixed integer linear programming problem.

In this work, we prove that the decision version of MRCP is NP-complete and formulate it as an integer programming problem with linear constraints. We investigate three solution methods, namely, a mixed integer programming approach based on linearization of the proposed integer formulation; binary search; and Newton's method. These methods are used to solve MRCP in stock market and wind energy graphs, as well as in instances from the 2^{nd} and 10^{th} DIMACS Implementation Challenges. The corresponding results are tabulated and compared.

The remainder of this chapter is organized as follows: In Section 4.2, we establish the computational complexity of the problem of interest and some related problems. In Sections 4.3 and 4.4 we formulate MRCP as an integer programming problem and describe the proposed solution methodologies. Section 4.5 describes graph instances used for testing and presents the results of computational experiments.

4.2 Computational Complexity

Given a simple undirected graph $G = (V, E)$, two non-negative rational weights, a_i and b_i, associated with each vertex $i \in V$, and a positive rational number c, the decision version of MRCP is to verify if there exists a maximal clique C in G such that $\frac{\sum_{i \in C} a_i}{\sum_{i \in C} b_i} \geq c$. The decision version of MAWCP is defined likewise.

Proposition 1. *The decision version of MRCP and the decision version of MAWCP are NP-complete.*

PROOF. Clearly, the decision version of MRCP is in the class NP. To prove NP-completeness, we reduce two different NP-complete problems to MRCP. First, we use a reduction from the minimum maximal clique problem, i.e. the problem of finding the maximal clique in a graph with the minimum number of vertices. This problem is equivalent to the minimum independent dominating set problem in the complement graph, the decision version of which is known to be NP-complete [26].

Let $G = (V, E)$ with $V = \{1, \ldots, n\}$ be an instance of the minimum maximal clique problem. Construct an instance $G' = (V', E')$ of MRCP with $V' = \{v_i : i \in V\} \cup \{v_0\}$, $E' = \{(v_i, v_j) : (i, j) \in E\} \cup \{(v_i, v_0) : i \in V\}$, and $a_i = 1 \; \forall i \in \{0, \ldots, n\}$; $b_0 = 1$; $b_i = 2 \; \forall i \in \{1, \ldots, n\}$.

Consider two maximal cliques, C_1 and C_2, in G', with $|C_1| = p{+}1$ and $|C_2| = q{+}1$, where $p < q$. Since v_0 is connected to every other vertex in G', it belongs to every maximal clique, i.e., $v_0 \in C_1$ and $v_0 \in C_2$. The MRCP objective value for C_1 is $f_1 = \frac{\sum_{i:v_i \in C_1} a_i}{\sum_{i:v_i \in C_1} b_i} = \frac{p+1}{2p+1}$, and that for C_2 is $f_2 = \frac{\sum_{i:v_i \in C_2} a_i}{\sum_{i:v_i \in C_2} b_i} = \frac{q+1}{2q+1}$. We have

$$f_1 - f_2 = \frac{p+1}{2p+1} - \frac{q+1}{2q+1} = \frac{q-p}{(2p+1)(2q+1)} > 0.$$

Note that any maximal clique in G' with vertex v_0 removed corresponds to a maximal

clique in G. Therefore, G has a maximal clique of size p or less *iff* G' has a maximal clique C with $\frac{\sum_{i:v_i \in C} a_i}{\sum_{i:v_i \in C} b_i} \geq c$, where $c = \frac{p+1}{2p+1}$. Thus, the decision version of MRCP is NP-complete.

Using the same construction G' with $a_0 = 2$, $a_i = 1 \; \forall i \in \{1, \ldots, n\}$; $b_i = 1 \; \forall i \in \{0, \ldots, n\}$, G has a maximal clique of size p or less *iff* G' has a maximal clique with $\frac{\sum_{i:v_i \in C} a_i}{\sum_{i:v_i \in C} b_i} \geq c$, where $c = 1 + \frac{1}{p+1}$. This shows that MAWCP is NP-complete (which also implies that MRCP is NP-complete and provides another way of solving the minimum maximal clique problem using methods for MRCP).

Note that if we set $a_0 = 0.5$ instead of $a_0 = 2$ above, it is easy to see that G has a clique of size at least p *iff* G' has a maximal clique C with $\frac{\sum_{i:v_i \in C} a_i}{\sum_{i:v_i \in C} b_i} \geq c$, where $c = 1 - \frac{0.5}{p+1}$, thus we obtain an alternative reduction from the maximum clique problem. The same arguments can be used to show that the decision version of the minimization problem, which asks to find a maximal clique with the minimum value of the fractional objective, is also NP-complete. □

To construct a feasible solution to MRCP it is sufficient to find any maximal clique in G, which can be easily done using, e.g., a simple greedy algorithm. Next we show that given an arbitrary feasible solution for MRCP finding a better solution is a hard problem. Subsequently, the problem Π_1 is defined as follows.

Π_1: Given a simple undirected graph $G = (V, E)$, and two non-negative rational weights, a_i and b_i, associated with each vertex $i \in V$, do there exist two maximal cliques C_1 and C_2 in G such that $\lvert \frac{\sum_{i \in C_1} a_i}{\sum_{i \in C_1} b_i} - \frac{\sum_{j \in C_2} a_j}{\sum_{j \in C_2} b_j} \rvert > 0$.

Proposition 2. *Problem* Π_1 *is NP-complete.*

PROOF. Problem Π_1 is clearly in NP. To show NP-completeness, we use a reduction from the following problem. A well-covered graph is a graph in which every maximal

independent set is a maximum independent set. The problem of recognizing a well-covered graph is known to be NP-hard [15, 53]. Let $G = (V, E)$ with $V = \{1, \ldots, n\}$ be an instance of the problem of recognizing a well-covered graph. Construct an instance $G' = (V', E')$ of Π_1 with $V' = \{v_i : i \in V\} \cup \{v_0\}$. For $i, j \in V$, an edge exists between v_i and v_j in G' *iff* there is *no* edge between vertices i and j in G. In addition, vertex v_0 is adjacent to all vertices in $V' \setminus \{v_0\}$. Set $a_0 = 2$; $a_i = 1\ \forall i \in \{1, \ldots, n\}$; and $b_i = 1\ \forall i \in \{0, \ldots, n\}$. The MRCP objective function value for any maximal clique of size $p + 1$ in G' is $\frac{p+2}{p+1}$. Hence, the absolute value of the difference between the MRCP objective function values of two maximal cliques is non-zero *iff* they have different cardinalities, i.e., *iff* G is not well-covered. Therefore, two maximal cliques C_1 and C_2 with $|\frac{\sum_{i:v_i \in C_1} a_i}{\sum_{i:v_i \in C_1} b_i} - \frac{\sum_{j:v_j \in C_2} a_j}{\sum_{j:v_j \in C_2} b_j}| > 0$ exist in G' *iff* G is not well-covered. □

Even though determining whether two feasible solutions with different MRCP objective function values exist in a graph is a hard problem in general, to prove that the answer is positive it suffices to find two maximal cliques with different objective values. Assuming that we are dealing with an instance of MRCP such that maximal cliques with different objective values are known to exist, an interesting question is how to find the *minimum* difference in the objective values that two maximal cliques can have. This difference could be used as a part of a stopping criterion ensuring an optimal solution in the methods proposed in Sections 4.4.2 and 4.4.3. Next, we show that obtaining this minimum difference is a hard problem. The formal definition of the problem, which is henceforth referred to as problem Π_2, is as follows.

Π_2: Given a simple undirected graph $G = (V, E)$ with the set $V = \{1, \ldots, n\}$ of vertices, where each vertex $i \in V$, is assigned two non-negative rational weights, a_i and b_i, find the least non-zero value of the quantity $|\frac{\sum_{i \in C_1} a_i}{\sum_{i \in C_1} b_i} - \frac{\sum_{j \in C_2} a_j}{\sum_{j \in C_2} b_j}|$, where C_1 and C_2 are any two maximal cliques in G, assuming that such a quantity exists.

Subsequently, the decision version of Π_2 is the following: Given a simple undirected graph $G = (V, E)$, two non-negative weights, a_i and b_i, associated with each vertex $i \in V$, and a positive rational number k, do there exist two maximal cliques C_1 and C_2 in G such that $|\frac{\sum_{i \in C_1} a_i}{\sum_{i \in C_1} b_i} - \frac{\sum_{j \in C_2} a_j}{\sum_{j \in C_2} b_j}| \leq k$?

Proposition 3. *The decision version of* Π_2 *is NP-complete.*

PROOF. The decision version of Π_2 is clearly in the class NP. To prove NP-completeness, a reduction from the maximum clique problem is used. Let $G = (V, E)$ with $V = \{1, \ldots, n\}$ be an instance of the maximum clique problem. Construct an instance $\tilde{G} = (\tilde{V}, \tilde{E})$ of Π_2 with $\tilde{V} = V' \cup V''$ and $\tilde{E} = E' \cup E''$, where $V' = \{v_i : i \in V\} \cup \{v_0\}$, $V'' = \{v_i : i = n+1, \ldots 2n\} \cup \{v_{2n+1}\}$; and $E' = \{(v_i, v_j) : (i, j) \in E\} \cup \{(v_0, v_i) : i \in V\}$, $E'' = \{(v_{n+i}, v_{n+j}) : (i, j) \in E\} \cup \{(v_{2n+1}, v_{n+i}) : i \in V\}$. Therefore, graph $\tilde{G}$ has two identical components defined by subsets of vertices V' and V'' with no edge between them. For graph $\tilde{G}$, set $a_i = 1 \; \forall i \in \{1, \ldots, 2n\}$, $a_0 = 2$, $a_{2n+1} = 1.5$, $b_i = 1 \; \forall i \in \{1, \ldots, 2n\}$, $b_0 = 1$ and $b_{2n+1} = 0.5$. Then the quantity of interest is

$$r(C_1, C_2) = \left| \frac{\sum_{i: v_i \in C_1} a_i}{\sum_{i: v_i \in C_1} b_i} - \frac{\sum_{j: v_j \in C_2} a_j}{\sum_{j: v_j \in C_2} b_j} \right|, \tag{22}$$

where C_1 and C_2 are any two maximal cliques in $\tilde{G}$. Let G' and G'' be the subgraphs of $\tilde{G}$ induced by V' and V'', respectively. Then G' and G'' are isomorphic and there is a one-to-one correspondence between maximal cliques in G' and G''. Hence, it can be

shown that $r(C_1, C_2)$ is minimized when $C_1 \subset V'$ and $C_2 \subset V''$ are maximum cliques of the same size in G' and G'', respectively. Then the least value that $r(C_1, C_2)$ can take is $r = \frac{0.5}{(p+1)(p+0.5)}$, where $p+1$ is the size of a maximum clique in G', which is the same as saying that p is the size of a maximum clique in G. Therefore, there exist two maximal cliques C_1 and C_2 in $\tilde{G}$ such that $r(C_1, C_2) \leq \frac{0.5}{(p+1)(p+0.5)}$ *iff* there exists a clique of size at least p in G. □

4.3 Integer Programming Formulation

The maximum ratio clique problem can be formulated as follows:

$$\text{maximize} \quad \frac{\sum_{i=1}^{n} a_i x_i}{\sum_{i=1}^{n} b_i x_i} \tag{23}$$

$$\text{subject to} \quad x_i + x_j \leq 1, \qquad \forall (i,j) \notin E, i \neq j \tag{24}$$

$$\sum_{i=1}^{n} (1 - a_{ij}) x_i \geq 1, \qquad \forall j \in V \tag{25}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in V, \tag{26}$$

where $\forall i \in V, x_i \in \{0,1\}$ is a decision variable indicating whether i is a part of the solution clique; a_{ij} are the elements of the adjacency matrix for graph G, that is $a_{ij} = 1$ if vertex i and vertex j are connected by an edge and 0 otherwise. Note that for the maximum weight clique problem, the objective function and the non-negativity of the weights force the optimal solution to the problem to be a maximal clique. On the other hand, for MRCP, we need to include additional constraints (25) to guarantee that the solution is a maximal clique. Otherwise, a single vertex i with maximum ratio a_i/b_i would trivially be an optimal solution for the above integer program.

4.4 Solution Methods

In this section, we propose three solution methods for MRCP, which will be compared in terms of their computational efficacy in Section 4.5.

4.4.1 Linearization

We use the formulation (23)–(26) to develop a mixed integer linear programming (MILP) formulation for MRCP. We follow the linearization framework proposed by Wu [64]. Namely, we introduce new variables $y = \frac{1}{\sum_{i=1}^{n} b_i x_i}$ and $z_i = y x_i,\ \forall i \in V$ and replace the quadratic expression for z_i with the following four linear constraints:

$$z_i \leq U x_i, \qquad \forall i \in V \tag{27}$$

$$z_i \geq L x_i, \qquad \forall i \in V \tag{28}$$

$$z_i \leq y - L(1 - x_i), \qquad \forall i \in V \tag{29}$$

$$z_i \geq y - U(1 - x_i), \qquad \forall i \in V, \tag{30}$$

where L and U are the upper and lower bounds on y, respectively. For MRCP, we can use L and U values given by

$$L = \frac{1}{\sum_{i=1}^{n} b_i} \text{ and } U = \frac{1}{\min_{1 \leq i \leq n}\{b_i\}},$$

respectively. The resulting MILP formulation is:

$$\text{maximize} \quad \sum_{i=1}^{n} a_i z_i \tag{31}$$

$$\text{subject to} \quad x_i + x_j \leq 1, \quad \forall (i,j) \notin E, i \neq j \tag{32}$$

$$\sum_{i=1}^{n} (1 - a_{ij}) x_i \geq 1, \qquad \forall j \in V \tag{33}$$

$$\sum_{i=1}^{n} b_i z_i = 1 \tag{34}$$

$$z_i \leq U x_i, z_i \geq L x_i, \quad \forall i \in V \tag{35}$$

$$z_i \leq y - L(1 - x_i), z_i \geq y - U(1 - x_i), \quad \forall i \in V \tag{36}$$

$$x_i \in 0, 1; \; L \leq y \leq U; \; z_i \geq 0, \quad \forall i \in V. \tag{37}$$

The first proposed solution method uses the MILP (31)–(37) in conjunction with a modern MILP solver.

4.4.2 Binary Search

Binary search has been used as a method to solve fractional programming problems by Lawler [39] and Ibaraki [30]. We adopt this method for MRCP as follows. For $\lambda \in \mathbb{R}_+$, denote by $P(\lambda)$ the optimal objective function value of the following binary program:

$$Q(\lambda) : \text{maximize} \quad \sum_{i=1}^{n} a_i x_i - \lambda (\sum_{i=1}^{n} b_i x_i) \tag{38}$$

$$\text{subject to} \quad x_i + x_j \leq 1, \quad \forall (i,j) \notin E, i \neq j \tag{39}$$

$$\sum_{i=1}^{n} (1 - a_{ij}) x_i \geq 1, \qquad \forall j \in V \tag{40}$$

$$x_i \in \{0, 1\}, \quad \forall i \in V \tag{41}$$

For MRCP, $P(\lambda)$ is convex, piece-wise linear and strictly decreasing. Let $C(\lambda)$ denote the maximal clique corresponding to the solution to $Q(\lambda)$. For a given value of λ, if $P(\lambda) > 0$, then for the maximal clique $C(\lambda)$, we have $\lambda < \frac{\sum_{i \in C(\lambda)} a_i}{\sum_{i \in C(\lambda)} b_i}$. On the other hand, if $P(\lambda) < 0$, then $\lambda > \frac{\sum_{i \in C} a_i}{\sum_{i \in C} b_i}$ for any maximal clique C in G. Therefore, the binary search algorithm seeks to find an approximation to λ^* such that $P(\lambda^*) = 0$.

Let $\tilde{C}$ be any maximal clique in G. Then $\frac{\sum_{i \in \tilde{C}} a_i}{\sum_{i \in \tilde{C}} b_i}$ gives a lower bound of the optimal ratio value for MRCP. Such $\tilde{C}$ can be found by selecting the vertex with the maximum value of the ratio of its weights and adding its neighbors in the decreasing order of the ratio of their weights while ensuring that the chosen set of vertices still forms a clique. An upper bound is given by $\max_{1 \leq i \leq n} \frac{a_i}{b_i}$. Let $\lambda_1 = \frac{\sum_{i \in \tilde{C}} a_i}{\sum_{i \in \tilde{C}} b_i}$ and $\lambda_2 = \max_{1 \leq i \leq n} \frac{a_i}{b_i}$. Then $\frac{\sum_{i \in C(\lambda_1)} a_i}{\sum_{i \in C(\lambda_1)} b_i}$ gives an improved lower bound (LB) and $\frac{\lambda_2 P(\lambda_1) - \lambda_1 P(\lambda_2)}{P(\lambda_1) - P(\lambda_2)}$ gives an improved upper bound (UB) for the root of $P(\lambda)$ [30]. At each iteration of the binary search algorithm, $\lambda = \frac{LB+UB}{2}$ is computed and $P(\lambda)$ is found along with the corresponding maximal clique solution $C(\lambda)$. For a given $\epsilon > 0$, if $UB - LB < \epsilon$, then the algorithm returns $C(\lambda)$ and the corresponding objective function value as an ϵ-approximation of the optimal objective value λ^* of MRCP. Otherwise, if $P(\lambda) > 0$, then the algorithm reassigns the value of λ_1 to be equal to λ and the corresponding values of LB and UB are updated. On the other hand, if $P(\lambda) < 0$, then the value of λ_2 is reassigned to be equal to λ and the value of UB is updated. The procedure is summarized in Algorithm 4.

It should be noted that the upper bound on the MRCP objective function value given by $\max_{1 \leq i \leq n} \frac{a_i}{b_i}$ might be of arbitrarily poor quality. For example, consider an instance of MRCP given by a graph $G = (\{v_1, v_2\}, (v_1, v_2))$ with the weights $a_1 = a$, $b_1 = 1$, $a_2 = 1$ and $b_2 = a$, where a is an arbitrarily large positive integer. Then the MRCP objective function value is $\frac{a+1}{a+1} = 1$, while the upper bound is a.

Algorithm 4 Binary search algorithm for MRCP

Input: $G = (V, E)$, $a_i, b_i \quad \forall i \in V$, any maximal clique $\tilde{C}$ in G, $\epsilon > 0$
Output: A maximal clique $C(\lambda_1)$ and $\lambda = (\sum_{i \in C(\lambda_1)} a_i)/(\sum_{i \in C(\lambda_1)} b_i)$ such that $\lambda^* - \lambda < \epsilon$, where λ^* is the optimal objective function value for MRCP

$\lambda_1 = \frac{\sum_{i \in \tilde{C}} a_i}{\sum_{i \in \tilde{C}} b_i}$
$\lambda_2 = \max_{1 \le i \le n} \frac{a_i}{b_i}$
Compute $P(\lambda_1)$, $C(\lambda_1)$, $P(\lambda_2)$ and $C(\lambda_2)$ by solving $Q(\lambda_1)$ and $Q(\lambda_2)$ in (38)–(41)
$LB = \frac{\sum_{i \in C(\lambda_1)} a_i}{\sum_{i \in C(\lambda_1)} b_i}$
$UB = \frac{\lambda_2 P(\lambda_1) - \lambda_1 P(\lambda_2)}{P(\lambda_1) - P(\lambda_2)}$
$\lambda = \lambda_1$, $P(\lambda) = P(\lambda_1)$ and $C(\lambda) = C(\lambda_1)$
Set the number of iterations $k = 1$
while $UB - LB \ge \epsilon$ **do**
  $\lambda = \frac{LB+UB}{2}$
  Compute $P(\lambda)$ and $C(\lambda)$
  if $P(\lambda) \ge 0$ **then**
    $\lambda_1 = \lambda$ and $P(\lambda_1) = P(\lambda)$
    $LB = \frac{\sum_{i \in C(\lambda_1)} a_i}{\sum_{i \in C(\lambda_1)} b_i}$
    $UB = \frac{\lambda_2 P(\lambda_1) - \lambda_1 P(\lambda_2)}{P(\lambda_1) - P(\lambda_2)}$
  else
    $\lambda_2 = \lambda$ and $P(\lambda_2) = P(\lambda)$
    $UB = \frac{\lambda_2 P(\lambda_1) - \lambda_1 P(\lambda_2)}{P(\lambda_1) - P(\lambda_2)}$
    $k = k + 1$
  end if
end while
return $C(\lambda)$, $\lambda = (\sum_{i \in C(\lambda)} a_i)/(\sum_{i \in C(\lambda)} b_i)$

The results of experiments with this algorithm are given in Section 4.5. The linear convergence of the binary search method for general fractional programming problems is shown by Ibaraki [30].

4.4.3 Newton's Method

Newton's Method for fractional programming problems was proposed by Isbell and Marlow [31] and has been adapted to MRCP here. We define $\tilde{C}$, $P(\lambda)$ and $C(\lambda)$ for $\lambda \in \mathbb{R}_+$, as in Section 4.4.2. The lower bound (λ_1) to MRCP is then given by $\frac{\sum_{i\in\tilde{C}} a_i}{\sum_{i\in\tilde{C}} b_i}$. We set the initial value of LB to be equal to λ_1. At each iteration of Newton's method, the values of $P(\lambda_1)$ and $C(\lambda_1)$ are computed. For a given $\epsilon > 0$, if $P(\lambda_1) < \epsilon$ then $C(\lambda_1)$ and the corresponding objective function value are returned. If $P(\lambda_1) > \epsilon$, then λ_1 is reset to the value $\frac{\sum_{i\in C(\lambda_1)} a_i}{\sum_{i\in C(\lambda_1)} b_i}$. The algorithm is summarized in Algorithm 5. The results of computational experiments with Algorithm 5 are given in Section 4.5. The quadratic rate of convergence of Newton's method for general fractional programs is shown by Ibaraki [30].

Algorithm 5 Newton's Method for the Maximum Ratio Maximal Clique Problem

---

Input: $G = (V, E)$, $a_i, b_i \quad \forall i \in V$, any maximal clique $\tilde{C}$ in G, $\epsilon > 0$
Output: A maximal clique $C(\lambda_1)$ and $\lambda = (\sum_{i\in C(\lambda_1)} a_i)/(\sum_{i\in C(\lambda_1)} b_i)$ such that $\lambda^* - \lambda < \epsilon/(\min_{i\in V} b_i)$, where λ^* is the optimal objective function value for MRCP

$\lambda_1 = \frac{\sum_{i\in\tilde{C}} a_i}{\sum_{i\in\tilde{C}} b_i}$
Compute $P(\lambda_1)$ and an optimal maximal clique solution $C(\lambda_1)$ of (38)–(41)
Set the number of iterations $k = 1$
while $P(\lambda_1) \geq \epsilon$ **do**
  $\lambda_1 = \frac{\sum_{i\in C(\lambda_1)} a_i}{\sum_{i\in C(\lambda_1)} b_i}$
  Compute $P(\lambda_1)$ and $C(\lambda_1)$
  $k = k + 1$
end while
return $C(\lambda_1)$, $\lambda = (\sum_{i\in C(\lambda_1)} a_i)/(\sum_{i\in C(\lambda_1)} b_i)$

Note that Algorithm 5 returns $C(\lambda_1)$, where λ_1 is such that $0 \leq P(\lambda_1) < \epsilon$, i.e.,

$$\sum_{i \in C(\lambda_1)} a_i - \lambda_1 \sum_{i \in C(\lambda)} b_i < \epsilon.$$

Let C^* be a maximum clique that solves MRCP to optimality. Then

$$(\sum_{i \in C^*} a_i)/(\sum_{i \in C^*} b_i) \geq \lambda \geq \lambda_1$$

and

$$\sum_{i \in C^*} a_i - \lambda_1 \sum_{i \in C^*} b_i \leq \sum_{i \in C(\lambda_1)} a_i - \lambda_1 \sum_{i \in C(\lambda_1)} b_i < \epsilon,$$

implying

$$0 \leq (\sum_{i \in C^*} a_i)/(\sum_{i \in C^*} b_i) - \lambda \leq (\sum_{i \in C^*} a_i)/(\sum_{i \in C^*} b_i) - \lambda_1 < \epsilon/(\sum_{i \in C^*} b_i) < \epsilon/(\min_{i \in V} b_i).$$

We could scale $b_i, i \in V$ to obtain an equivalent instance of MRCP with $\min_{i \in V} b_i = 1$, thus obtaining an ϵ-approximation algorithm with respect to the objective function of MRCP.

Thus, Algorithms 4 and 5 are ϵ-approximation algorithms. The results from propositions 2 and 3 show that the value of ϵ such that the algorithms can be claimed to be exact cannot be found in polynomial time, unless P=NP.

4.5 Results of Computational Experiments

In this section we present the results of computational experiments with the proposed methods for solving MRCP. We start by describing the test instances used in the experiments.

4.5.1 Description of Test Instances

The instances used in the computational experiments are constructed as follows.

Set A and Set B. We generated uniform random graphs, where each edge exists with probability 0.5, with both weights as integers between 1 and 100 following a discrete uniform distribution. For Set A, no additional constraint was imposed on these weights. However, in practical applications like the stock market graph, the two weights at each vertex may be highly correlated. Therefore, in Set B, the two weights of each vertex were constrained to be within 1.5% of each other.

Set C. We constructed this set of instances using stock market data for 500 financial instruments over a 500-day period, as described in [9]. We calculated the correlation of price fluctuations for each pair of stocks over the considered period. An edge exists between two vertices *iff* this correlation is below a threshold value set at 0 for the purpose of experiments. The first and second weights for each node are the prices of stocks of the corresponding financial instrument on the last and first day, respectively. These represent the selling and buying prices of the financial instrument over this time period.

Set D. Another application considered was establishment of wind turbines. We used wind energy information from [1] to determine 250 prospective locations for wind energy farms. The historical wind speed data at these locations at heights of 80 and 100 meters are available at the same source. Each location results in two vertices in the wind turbine graph. The first vertex corresponds to the construction of a 80 meter high wind turbine at the location and the second corresponds to the construction of a 100 meter high wind turbine. The first weight of each vertex is the average wind speed at the corresponding location and height. The second weight for each node is proportional to the sum of the installed capital cost and the annual

operating cost for the corresponding height of the wind turbine. In this case, the weight of the 80 meter high turbine was set to 1 and that of the 100 meter high turbine was set to be equal to the ratio of the net costs of the 100 meter high turbine to that of the 80 meter high turbine. The capital and operating costs are obtained from [62]. Two nodes are connected if the wind speeds at the two locations are negatively correlated (i.e., the correlation threshold is 0) in the historical data. This analysis was conducted for the data from 3 years: 2004, 2005 and 2006. In practice, the wind speeds over the years can be averaged to form an instance of MRCP.

Set E. Observing that the proof of Proposition 1 gives a simple one-to-one correspondence between optimal solutions of the minimum maximal clique problem and MRCP, we solved the minimum maximal clique problem on instances from the 2^{nd} and 10^{th} DIMACS Implementation Challenges using the first reduction in the proof of Proposition 1. Based on the construction of the instance of MRCP from the corresponding instance of the minimum maximal clique problem, the number of vertices in the constructed graph and in its maximum ratio clique are one more than those in the original graph and its minimum maximal clique, respectively.

Set F. Finally, we solved the maximum ratio maximal clique problem on instances from the 2^{nd} and 10^{th} DIMACS Implementation Challenges. For a graph with vertex set V, the first and second weights for vertex i were assigned to be equal to i and $|V| - i + 1$, respectively, for $i \in V$.

Table 29 summarizes the basic characteristics of the instances used. In particular, it provides the name of each instance, its number of vertices, number of edges, density, the optimal objective function value, and the size of a maximal clique representing an optimal solution. The entries reported in the last two columns were obtained using the proposed solution methods as described next.

4.5.2 Comparison of Results

We applied the three methods described in Section 4.4 to solve MRCP on the considered graph instances using a DELL OPTIPLEX 960 computer with INTEL(R) CORE(TM) 2 QUAD 3 GHZ processor and 8 GB of RAM. We implemented the algorithms using Microsoft Visual Studio 2008 and CPLEX 11 as the MILP solver. We set a time limit of 4 hours for the linearization method. For the binary search and Newton's method, the time limit for each iteration was 3 hours and the next iteration was begun only if the clock time did not exceed 3 hours. In addition, $\epsilon = 10^{-3}$ was used in the stopping criterion. Results reported are only for the cases where at least one of the methods returned a solution within the given time limits.

Table 30 compares the results obtained using the solution methodologies proposed in Section 4.4 in terms of CPU time in seconds. In addition, the number of iterations required for convergence for the binary search and Newton's method are tabulated under the columns "Steps". In this table "TE" stands for unavailability of results due to the time limit being exceeded and "OM" stands for unavailability of results due to *CPLEX out of memory* error. It should be noted that the binary search requires the solution of problem $Q(\lambda)$ for for the lower and upper bounds in the first iteration while Newton's method algorithm requires this only for the lower bound. Therefore, even with the same or smaller number of steps, the performance of the binary search method might be worse than that of the Newton's method. Below we summarize our observations concerning the results of experiments for each considered set of test instances.

Sets A and B. The densities of these instances in Table 29 are close to 50% as expected from the construction. The objective function in all instances in Set B is close to 1 since the weights are constrained to be close to each other. In Table 30,

we observe that for all these instances, Newton's method and binary search perform better than the linearization method. The difference becomes more pronounced as the size of the instances increases. For some instances, the linearization method did not return an optimal solution within the time limit. The performance of Newton's method and binary search were comparable to each other on the considered random graphs.

Set C. The high values of the ratio of the sum of selling prices to the sum of the buying prices in the optimal solution in Table 29 show that at least some of the financial instruments corresponding to the vertices of the optimal maximal clique themselves have a high value of the ratio between their weights. For these instances, Newton's method performed better than binary search, which in its turn performed better than linearization as seen in Table 30. The time required to solve these instances was considerably lower than the time required to solve the instances with 500 vertices in the first two sets of instances. This can be explained by observing that these graphs have a lower edge density than the uniform random graphs used.

Set D. As can be seen from the description in Table 29, these graphs have a very low edge density. The correlation threshold can be varied to obtain a higher density and subsequently a higher clique size in the optimal solution. The high values of the objective function result from using the relative costs as the second weights of the vertices instead of the actual costs. The comparison of solution methods on these graphs in Table 30 shows the same trend as in the stock market graphs. The performance of Newton's method is again the best followed by the binary search and linearization methods in that order. The time required to obtain the solution for these instances is lower compared to the time required for other instances with 500 vertices seen up till now. This again corresponds to the decrease in edge density in these instances.

Set E. Recall that based on the construction of an instance of MRCP from the corresponding instance of the minimum maximal clique problem, the number of vertices in a maximum ratio clique is one more than that in the corresponding minimum maximal clique in the original graph. These are tabulated in the column "Clique Size" in Table 29. The cardinality of the minimum maximal clique in these instances is always 2 for the instances listed between *karate* and *email*. This is consistent with the low density of these graphs. The same trend as before is observed in the performance of the three methods in Table 30. Newton's method and binary search are comparable for this set of instances. However, Newton's method is seen to be more robust than the binary search, since there are instances in which a solution is obtained within the time limit for Newton's method, but not for binary search. Both methods in general perform better than linearization. However, there are some exceptions, e.g., *c-fat500-1* and *c-fat500-10*.

Set F. Since the average of the numerator and denominator weights over all the vertices is equal in these instances, the objective function value in Table 29 is close to 1 in many cases. In general, for this set of instances Newton's method outperforms binary search and binary search outperforms linearization. However, we again observe exceptions in *c-fat* graphs and *johnson32-2-4* graph.

The number of iterations of Newton's method and binary search has not been seen to be very different from each other in general. However, for the instances for which binary search performs better, the number of steps was at least one less than that of the Newton's method. This is consistent with the number of times the problem $Q(\lambda)$ is solved in both methods. From the numerical experiments above, it can be concluded that Newton's method can be expected to perform better than binary search and linearization in many cases. However, graphs with a specific structure might arise, which make it easier for CPLEX to solve the linearized formulation

faster than expected. Similarly, if the lower and upper bounds are very effective, then binary search can perform better than Newton's method. For many instances of the DIMACS Implementation Challenges, the lower and upper bounds described in Section 4.4 were very tight. This is discerned from the number of iterations, which was 1 for many of these instances.

Table 29: Description of test instances used, optimal objective function values, and the corresponding maximal clique sizes.

#	$\|V\|$	$\|E\|$	Density (%)	Obj. Fn.	Clique Size
Set A: Uniform random graphs with uncorrelated weights					
random-1	100	2266	45.78	3.28	10
random-2	150	5212	46.64	4.69	8
random-3	200	10008	50.29	4.21	5
random-4	400	40786	51.11	4.83	7
random-5	500	63789	51.13	3.65	9
Set B: Uniform random graphs with constrained vertex ratio					
random-6	100	2655	53.64	1.15	13
random-7	150	5767	51.61	1.20	9
random-8	200	10220	51.36	1.19	10
random-9	400	38942	48.80	1.32	7
random-10	500	62444	50.06	1.37	9
Set C: Stock market graphs					
market-1	500	23116	18.53	21.21	4
market-2	500	26431	21.19	23.69	5
market-3	500	27704	22.21	13.92	4
market-4	500	25819	20.70	18.74	5

Table 29: (continued)

#	$\|V\|$	$\|E\|$	Density (%)	Obj. Fn.	Clique Size
market-5	500	26196	21.00	14.19	4
Set D: Wind energy graphs					
wind-2004	500	10277	8.24	94142.30	3
wind-2005	500	10516	8.43	94686.60	2
wind-2006	500	9681	7.76	98471.00	2
Set E: DIMACS instances for minimum maximal clique					
brock200_1	201	15034	74.80	0.53	9
brock 200_2	201	10076	50.13	0.56	5
brock200_3	201	12248	60.94	0.55	6
brock200_4	201	13289	66.11	0.54	7
c-fat200-1	201	1734	8.63	0.52	11
c-fat200-2	201	3435	17.09	0.51	23
c-fat200-5	201	8673	43.15	0.50	57
c-fat500-1	501	4959	3.96	0.52	13
c-fat500-2	501	9639	7.70	0.51	25
c-fat500-5	501	23691	18.91	0.50	63
c-fat500-10	501	47127	37.63	0.50	125
hamming6-2	65	1888	90.77	0.52	13
hamming6-4	65	768	36.92	0.60	3
hamming8-4	257	21120	64.20	0.56	5
johnson8-2-4	29	238	58.62	0.56	5
johnson8-4-4	71	1925	77.46	0.53	8
johnson16-2-4	121	5580	76.86	0.53	9

Table 29: (continued)

#	$\|V\|$	$\|E\|$	Density (%)	Obj. Fn.	Clique Size
johnson32-2-4	497	108376	87.93	0.52	17
keller4	172	9606	65.32	0.55	6
p_hat300-1	301	11233	24.88	0.57	4
p_hat300-2	301	22228	49.23	0.55	6
p_hat300-3	301	33690	74.62	0.53	10
p_hat500-1	501	32069	25.60	0.57	4
p_hat700-1	701	61699	25.15	0.57	4
p_hat1000-1	1001	123253	24.63	0.57	4
san200_0.7_1	201	14130	70.30	0.54	7
san200_0.7_2	201	14130	70.30	0.54	7
san200_0.9_1	201	18110	90.10	0.52	16
san200_0.9_2	201	18110	90.10	0.52	17
san200_0.9_3	201	18110	90.10	0.52	16
san400_0.5_1	401	40300	50.25	0.56	5
san400_0.7_2	401	56260	70.15	0.53	9
san1000	1001	251500	50.25	0.55	6
sanr200_0.7	201	14068	69.99	0.53	8
sanr200_0.9	201	18063	89.87	0.52	17
sanr400_0.5	401	40384	50.35	0.54	6
sanr400_0.7	401	56269	70.16	0.53	9
karate	35	112	18.82	0.60	3
dolphins	62	221	11.69	0.60	3
polbooks	106	546	9.81	0.60	3

Table 29: (continued)

#	$\|V\|$	$\|E\|$	Density (%)	Obj. Fn.	Clique Size
adjnoun	113	537	8.49	0.60	3
football	116	728	10.91	0.60	3
jazz	199	2940	14.92	0.60	3
celegans_metabolic	454	2478	2.41	0.60	3
email	1134	6584	1.02	0.60	3
Set F: DIMACS instances with $a_i = i, b_i = \|V\| - i + 1$					
brock200_1	200	14834	74.54	6.88	12
brock 200_2	200	9876	49.63	15.30	6
brock200_3	200	12048	60.54	13.89	8
brock200_4	200	13089	65.77	8.63	8
c-fat200-1	200	1534	7.71	1.22	10
c-fat200-2	200	3235	16.26	1.15	22
c-fat200-5	200	8473	42.58	1.02	58
c-fat500-1	500	4459	3.57	1.26	12
c-fat500-2	500	9139	7.33	1.07	26
c-fat500-5	500	23191	18.59	1.04	62
c-fat500-10	500	46627	37.38	1.01	126
hamming6-2	64	1824	90.48	1.65	16
hamming6-4	64	704	34.92	2.94	2
hamming8-4	256	20864	63.92	1.99	9
johnson8-2-4	28	210	55.56	1.15	4
johnson8-4-4	70	1855	76.81	1.41	7
johnson16-2-4	120	5460	76.47	1.26	8

Table 29: (continued)

#	$\|V\|$	$\|E\|$	Density (%)	Obj. Fn.	Clique Size
johnson32-2-4	496	107880	87.88	18.59	16
keller4	171	9435	64.91	4.15	7
keller5	776	225990	75.15	4.53	15
p_hat300-1	300	10933	24.38	25.76	4
p_hat300-2	300	21928	48.89	7.51	13
p_hat300-3	300	33390	74.45	7.23	16
p_hat500-1	500	31569	25.31	19.66	4
p_hat700-1	700	60999	24.93	21.08	4
p_hat1000-1	1000	122253	24.48	35.01	5
san200_0.7_1	200	13930	70.00	10.96	15
san200_0.7_2	200	13930	70.00	18.14	12
san200_0.9_1	200	17910	90.00	4.94	20
san200_0.9_2	200	17910	90.00	5.03	26
san200_0.9_3	200	17910	90.00	4.55	21
san400_0.5_1	400	39900	50.00	57.68	6
san400_0.7_1	400	55860	70.00	12.53	11
san400_0.7_2	400	55860	70.00	16.19	15
san400_0.7_3	400	55860	70.00	17.94	12
san1000	1000	250500	50.15	83.42	7
sanr200_0.7	200	13868	69.69	10.06	11
sanr200_0.9	200	17863	89.76	4.86	23
sanr400_0.5	400	39984	50.11	15.23	7
sanr400_0.7	400	55869	70.01	14.19	10

Table 29: (continued)

#	$\|V\|$	$\|E\|$	Density (%)	Obj. Fn.	Clique Size
karate	34	78	13.90	16.5	3
dolphins	62	159	8.41	11.6	2
polbooks	105	441	8.08	12.25	3
adjnoun	112	425	6.84	19.55	2
football	115	613	9.35	9.55	2
jazz	198	2742	14.06	7.29	2
celegans	453	2025	1.98	6.32	2
email	1133	5451	0.85	57.15	2

Table 30: Comparison of the results of experiments using the proposed approaches on the graph instances described in Table 29.

Graph	$\|V\|$	Linearization	Binary Search		Newtons Method	
		Time	Time	Steps	Time	Steps
Set A: Uniform random graphs with uncorrelated weights						
random-1	100	8.3	1.0	2	0.8	2
random-2	150	3.4	2.1	2	2.2	3
random-3	200	60.6	5.3	3	4.9	3
random-4	400	4213.5	95.0	3	77.2	2
random-5	500	TE	937.5	3	929.0	3
Set B: Uniform random graphs with constrained vertex ratio						
random-6	100	2.9	0.6	3	0.2	2
random-7	150	80.7	4.5	4	4.6	4
random-8	200	267.7	25.2	4	10.3	3

Table 30: (continued)

Graph	$\|V\|$	Linearization	Binary Search		Newtons Method	
		Time	Time	Steps	Time	Steps
random-9	400	5642.4	159.1	2	183.9	4
random-10	500	TE	219.8	3	316.7	4
Set C: Stock market graphs						
market-1	500	2428.3	121.8	4	82.8	3
market-2	500	910.9	80.7	3	38.6	2
market-3	500	2527.2	117.3	4	68.5	3
market-4	500	1834.4	110.1	4	64.5	3
market-5	500	1294.5	79.9	2	41.2	2
Set D: Wind energy graphs						
wind-2004	500	484.5	85.8	3	42.5	2
wind-2005	500	161.0	93.2	3	70.6	3
wind-2006	500	98.8	42.3	1	41.6	2
Set E: DIMACS instances for minimum maximal clique						
brock200_1	201	TE	1420.3	1	1117.9	2
brock 200_2	201	5623.8	104.2	1	81.6	2
brock200_3	201	TE	465.7	1	363.2	2
brock200_4	201	TE	764.6	1	784.4	2
c-fat200-1	201	429.8	20.7	1	21.9	2
c-fat200-2	201	347.2	13.5	1	13	2
c-fat200-5	201	241.2	12.3	1	12.3	2
c-fat500-1	501	119.2	447.1	1	397.8	2
c-fat500-2	501	347.2	348.3	1	337.6	2

Table 30: (continued)

Graph	$\|V\|$	Linearization	Binary Search		Newtons Method	
		Time	Time	Steps	Time	Steps
c-fat500-5	501	241.2	217	1	218.7	2
c-fat500-10	501	119.2	143	1	152.7	2
hamming6-2	65	57.5	0.4	1	0.4	2
hamming6-4	65	2.7	< 0.1	1	< 0.1	2
hamming8-4	257	TE	1.6	1	1.5	2
johnson8-2-4	29	< 0.1	< 0.1	1	< 0.1	1
johnson8-4-4	71	49.4	3.2	1	2.7	2
johnson16-2-4	121	0.7	0.4	1	0.3	1
johnson32-2-4	497	TE	1659.5	1	1186.5	1
keller4	172	5640.4	26.6	1	28.1	2
p_hat300-1	301	8002.8	99.9	1	98.7	2
p_hat300-2	301	TE	205.7	1	205	2
p_hat300-3	301	TE	1547.5	1	1787.1	2
p_hat500-1	501	TE	889.7	1	660.9	2
p_hat700-1	701	TE	6701	1	6944	2
p_hat1000-1	1001	TE	TE	TE	17094.3	2
san200_0.7_1	201	TE	348.2	1	448.9	2
san200_0.7_2	201	TE	104.5	1	106.8	2
san200_0.9_1	201	TE	3888.5	1	3998.8	2
san200_0.9_2	201	TE	3436.2	1	4245.6	2
san200_0.9_3	201	TE	10480	1	12779.9	2
san400_0.5_1	401	TE	1642.5	1	1672.8	2

Table 30: (continued)

Graph	$\|V\|$	Linearization	Binary Search		Newtons Method	
		Time	Time	Steps	Time	Steps
san400_0.7_2	401	TE	TE	TE	21596.8	2
san1000	1001	TE	TE	TE	21596.7	2
sanr200_0.7	201	TE	654.1	1	847.9	2
sanr200_0.9	201	TE	TE	TE	7519.1	2
sanr400_0.5	401	TE	TE	TE	TE	TE
sanr400_0.7	401	TE	TE	TE	21596.6	2
karate	35	0.08	< 0.1	1	< 0.1	2
dolphins	62	0.64	< 0.1	1	< 0.1	2
polbooks	106	3.71	0.3	1	0.3	2
adjnoun	113	3.15	0.3	1	0.3	2
football	116	5.86	0.6	1	0.6	2
jazz	199	80.37	1.6	1	1.8	2
celegans	454	602.507	104.3	4	41.6	2
email	1134	TE	1479.6	1	1449.4	2
Set F: DIMACS instances with $a_i = i, b_i = \|V\| - i + 1$						
brock200_1	200	4047.9	23.2	3	10.1	3
brock 200_2	200	155.6	1.5	1	1.2	2
brock200_3	200	342.1	2.3	1	1.5	2
brock200_4	200	1341.6	16.1	3	4.1	3
c-fat200-1	200	33.5	32.6	1	32.7	2
c-fat200-2	200	35.1	48.3	1	67.2	2
c-fat200-5	200	21.5	44.3	1	28.5	1

Table 30: (continued)

Graph	$\|V\|$	Linearization	Binary Search		Newtons Method	
		Time	Time	Steps	Time	Steps
c-fat500-1	500	827.3	850	1	1253.2	2
c-fat500-2	500	1375.7	953.9	1	822.4	1
c-fat500-5	500	275.2	3251.6	1	6749.2	2
c-fat500-10	500	203.9	1569.8	1	2623.9	2
hamming6-2	64	44.4	3.5	4	0.8	3
hamming6-4	64	0.8	0.3	1	0.3	2
hamming8-4	256	TE	854.7	2	234.4	2
johnson8-2-4	28	0.2	0.1	1	0.0	2
johnson8-4-4	70	46.3	9.7	3	5.6	3
johnson16-2-4	120	2394.2	708.6	1	231.0	2
johnson32-2-4	496	7670.6	OM	OM	OM	OM
keller4	171	380.7	5.9	1	2.0	2
keller5	776	TE	TE	TE	10184.7	2
p_hat300-1	300	431.1	9.7	2	12.9	4
p_hat300-2	300	14296.6	107.8	3	25.6	2
p_hat300-3	300	TE	105.6	3	66.7	3
p_hat500-1	500	8768.1	129.6	3	66.8	3
p_hat700-1	700	TE	286.6	3	239.0	3
p_hat1000-1	1000	TE	654.3	1	482.0	2
san200_0.7_1	200	298.5	7.5	2	0.6	2
san200_0.7_2	200	222.4	0.5	1	0.4	1
san200_0.9_1	200	TE	17.4	2	9.3	3

Table 30: (continued)

Graph	$\|V\|$	Linearization	Binary Search		Newtons Method	
		Time	Time	Steps	Time	Steps
san200_0.9_2	200	TE	22.9	3	3.1	3
san200_0.9_3	200	TE	41.4	3	20.0	3
san400_0.5_1	400	218.3	6	1	5.4	2
san400_0.7_1	400	TE	128.3	1	24.7	2
san400_0.7_2	400	TE	40.6	1	1.9	1
san400_0.7_3	400	10986.6	13.9	3	5.0	3
san1000	1000	TE	179.8	1	176.7	2
sanr200_0.7	200	805.2	3.5	2	1.3	2
sanr200_0.9	200	TE	22.3	2	5.2	3
sanr400_0.5	400	8505.2	64.8	3	37.0	3
sanr400_0.7	400	TE	96.6	3	33.6	3
karate	34	0.1	< 0.1	1	< 0.1	1
dolphins	62	0.2	< 0.1	1	< 0.1	1
polbooks	105	1.1	0.3	1	0.2	2
adjnoun	112	0.6	0.3	1	0.3	2
football	115	2.4	1.2	2	1.4	3
jazz	198	60.7	11.4	3	13.8	4
celegans	453	463.6	116.6	4	71.7	3
email	1133	1476.9	1164.7	1	578.9	1

4.6 Conclusion

In this chapter, we introduce the fractional version of the maximum clique problem, the maximum ratio clique problem, which may find many interesting applications. We show that the decision version of this problem is NP-complete and propose three solution methods. In the first method, the fractional objective function is converted into a linear one by introducing additional variables and constraints. This linear formulation is then used in conjunction with a modern MILP solver. The other two methods, binary search and Newton's method are ϵ-approximate methods. We show that it is NP-hard to determine the value of ϵ which will ensure that these algorithms are exact. We performed numerical experiments on randomly generated instances, application-specific instances, and on standard graphs from the 2^{nd} and 10^{th} DIMACS Implementation Challenges. It was observed that most instances with up to 500 vertices can be solved using the proposed solution methods. In terms of CPU time, the results show a superior performance of Newton's method in most instances. However, there are cases where the linearization and/or binary search methods perform better. Most of the test instances we used in this chapter are available in public domain and could be used as benchmarks for evaluation of new methods.

5. CONCLUSIONS AND FUTURE WORK

Improvement of system efficiency is one of the main objectives of any organization. This system efficiency is quantified by certain performance measures. These performance measures can be improved upon by making changes to the input parameters that are under human control.

To make an informed decision on the changes to be made in these input parameters, the basic requirement is to know the relationship between the performance measure and the input parameters. In some systems, this first step itself becomes an extremely difficult problem. One set of systems that fall into this category are fork-join queueing networks. The performance measure in this context is the average response time in the system. The problem of response time estimation of fork-join queues has been around in literature for more than thirty years. This response time is the maximum of a set of random variables that are highly correlated. This correlation and consequently, the mean response time in fork-join queueing networks, has proved to be extremely hard to quantify. We propose expressions for the mean response time in terms of the job arrival and task service rates and the number of tasks that need to be completed for the job to be complete. The symmetric n-dimensional fork-join queueing system has been analyzed by many researchers. However, our estimation technique easily scores over all the approximations available in literature in terms of its simplicity, time and computing resources consumed, and most importantly its remarkable accuracy. We have been able to extend the estimation technique of the symmetric n-dimensional fork-join queue to more complex forms of fork-join queueing networks. To the best of our knowledge, no approximations exist in literature for these fork-join queues. Therefore, our contribution and its importance to this widely

applied set of queueing systems is crystal clear. Our expressions for the average response times are so simple that the appropriate input parameters can be estimated for the best possible performance of the system instantaneously in most cases.

Many exciting prospects for future research in the area of fork-join queues have opened up with the work in this dissertation. The most obvious and glamorous of these is providing a proof for the conjectures in Chapters 2 and 3. The simulations presented in this dissertation provide strong justification for the use of these conjectures in their present form in applications. However, obtaining a proof would be gratifying for any researcher. Apart from this, other and possibly easier research prospects are the following:

1. In the symmetric tandem fork-join queueing system, we consider the case where the sub-task service time is exponentially distributed. Since the expressions for the mean response time are accurate for general service times in the symmetric n-dimensional fork-join queueing system, therefore, it is highly likely that these results can be extended to the case of general service times in the tandem system as well. Extensive simulations need to be conducted to test this postulate.

2. In Chapter 3, we provide an expression that is strongly supported by simulations for the (n, k) fork-join queueing system. However, this expression cannot be used directly to estimate the mean response time of this system because the mean response time of a single task queue is dependent on the other queues in the system. Nonetheless, it might be possible to obtain tighter bounds on the mean response time of this single task queue when compared to that of the entire system. These bounds can be used to used in conjunction with the expression proposed in this dissertation to provide bounds on the system re-

sponse time. The bounds obtained using this procedure are likely to perform better than those available in literature.

3. It would be interesting to approximate the mean response times in acyclic fork-join queueing systems which are seen in the work of Baccelli et. al. [6] using the expressions for the symmetric tandem fork-join queueing system. There are no approximations available in literature for these systems. Therefore, if the approximations obtained using this technique prove to be fairly accurate, then this will be a significant contribution to the fork-join queueing literature.

In the second part of this dissertation, we move on to systems for which the performance measures are expressed in terms of ratios of functions of the input parameters. These input parameters represent limited resources. The constraints on these resources could potentially prohibit the use of one resource at the same time as another. Such systems are seen in application areas such as social network analysis, stock markets and establishment of wind farms. The problem of determining the set of input parameters that will maximize the performance measure is formulated as a combinatorial optimization problem known as the *maximum ratio clique problem*. We prove the complexity of this problem and compare solution methods to determine the one that performs best in terms of computing time. In this first piece of work dealing with the maximum ratio clique problem, we restrict the discussion to very basic solution approaches, leaving much room for future improvements, some of which are enumerated below:

1. The binary search and Newton's method algorithms require the solution to an integer programming problem at each iteration. A polyhedral study of the polytope of this problem might result in better performance of these methods. The valid inequalities for the maximum clique problem are also valid for this

integer programming problem. A preliminary run with some valid inequalities based on maximal independent sets shows promising results and provides a motivation for proceeding in this direction.

2. Exploiting alternative mixed integer linear programming reformulations of the fractional programming formulation could prove beneficial [55].

3. Developing effective heuristic methods for solving large-scale instances of the problem is another interesting research avenue to explore.

REFERENCES

[1] Nrel: Transmission grid integration - data and resources. `http://www.nrel.gov/electricity/transmission/data_resources.html`, 2012. Last accessed: October 2015.

[2] M. H. Ammar and S. B. Gershwin. Equivalence relations in queueing models of fork/join networks with blocking. *Performance Evaluation*, 10(3):233 – 245, 1989.

[3] F. Baccelli and Z. Liu. On the execution of parallel programs on multiprocessor systems–a queuing theory approach. *J. ACM*, 37(2):373–414, Apr. 1990.

[4] F. Baccelli and A. Makowski. Simple computable bounds for the fork-join queue. In *Proceedings of the Conference of Information Science Systems, John Hopkins University*, pages 436–441, 1985.

[5] F. Baccelli, A. M. Makowski, and A. Shwartz. The fork-join queue and related systems with synchronization constraints: Stochastic ordering and computable bounds. *Advances in Applied Probability*, 21(3):629–660, 1989.

[6] F. Baccelli, W. A. Massey, and D. Towsley. Acyclic fork-join queuing networks. *J. ACM*, 36(3):615–642, 1989.

[7] S. Banerjee, P. Gupta, and S. Shakkottai. Towards a queueing-based framework for in-network function computation. *Queueing Systems*, 72(3-4):219–250, 2012.

[8] A. Billionnet. Approximation algorithms for fractional knapsack problems. *Operations Research Letters*, 30(5):336–342, 2002.

[9] V. Boginski, S. Butenko, and P. Pardalos. Mining market data: A network approach. *Computers and Operations Research*, 33:3171–3184, 2006.

[10] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 1–74, Dordrecht, The Netherlands, 1999. Kluwer Academic Publishers.

[11] O. Boxma, G. Koole, and Z. Liu. Queueing-theoretic solution methods for models of parallel and distributed systems. In *Performance Evaluation of Parallel and Distributed Systems Solution Methods. CWI Tract 105 & 106*, 1996.

[12] R. Chandrasekaran. Minimal ratio spanning trees. *Networks*, 7(4):335–342, 1977.

[13] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. Raid: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[14] R. Chen, H. Zhang, and H. Hu. A fast simulation for thousands of general homogeneous fork/join queues. In D. Al-Dabass, A. Pantelous, H. Tawfik, and A. Abraham, editors, *International Conference on Intelligent Systems, Modelling and Simulation*, pages 300–305, 2010.

[15] V. Chvátal and P. Slater. A note on well-covered graphs. *Ann. Discrete Math.*, 55:179–182, 1993.

[16] H. Dai. Exact monte carlo simulation for fork-join networks. *Advances in Applied Probability*, 43(2):484–503, 2011.

[17] Y. Dallery, Z. Liu, and D. Towsley. Equivalence, reversibility, symmetry and concavity properties in fork-join queuing networks with blocking. *J. ACM*, 41(5):903–942, Sept. 1994.

[18] G. B. Dantzig, W. O. Blattner, and M. R. Rao. Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem. In P. Rosentlehl, editor, *Theory of Graphs*, pages 77–84, New York, 1967. Gordon and Breach.

[19] A. Dasdan and R. Gupta. Faster maximum and minimum mean cycle algorithms for system-performance analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(10):889–899, 1998.

[20] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

[21] A. Duda and T. Czachórski. Performance evaluation of fork and join synchronization primitives. *Acta Informatica*, 24(5):525–553, 1987.

[22] A. K. Erlang. The theory of probabilities and telephone conversations. *Nyt Tidsskrift for Matematik B*, 20(16):33–39, 1909.

[23] L. Flatto. Two parallel queues created by arrivals with two demands II. *SIAM Journal on Applied Mathematics*, 45(5):861–878, 1985.

[24] L. Flatto and S. Hahn. Two parallel queues created by arrivals with two demands I. *SIAM Journal on Applied Mathematics*, 44(5):1041–1053, 1984.

[25] B. Fox. Finding minimal cost-time ratio circuits. *Operations Research*, 17(3):546–551, 1969.

[26] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, New York, 1979.

[27] S. B. Gershwin. Assembly/disassembly systems: An efficient decomposition algorithm for tree-structured networks. *IIE Transactions*, 23(4):302–314, 1991.

[28] C. Gkantsidis, M. Ammar, and E. Zegura. On the effect of large-scale deployment of parallel downloading. In *Proceedings of The Third IEEE Workshop on Internet Applications*, pages 79–89, 2003.

[29] V. R. Guide Jr., G. C. Souza, and E. van der Laan. Performance of static priority rules for shared facilities in a remanufacturing shop with disassembly and reassembly. *European Journal of Operational Research*, 164(2):341 – 353, 2005.

[30] T. Ibaraki. Parametric approaches to fractional programs. *Mathematical Programming*, 26:345–362, 1983.

[31] J. R. Isbell and W. H. Marlow. Attrition games. *Naval Research Logistics Quarterly*, 3(1–2):71–94, 1956.

[32] G. Joshi, Y. Liu, and E. Soljanin. On the delay-storage trade-off in content download from coded distributed storage systems. *IEEE Journal on Selected Areas in Communications*, 32(5):989–997, 2014.

[33] R. M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978.

[34] B. Kemper and M. Mandjes. Mean sojourn times in two-queue fork-join systems: bounds and approximations. *OR spectrum*, 34(3):723–742, 2012.

[35] C. Kim and A. Agrawala. Analysis of the fork-join queue. *IEEE Transactions on Computers*, 38(2):250–255, Feb 1989.

[36] S.-S. Ko and R. F. Serfozo. Response times in M/M/s fork-join networks. *Advances in Applied Probability*, 36(3):pp. 854–871, 2004.

[37] A. Kumar and R. Shorey. Performance analysis and scheduling of stochastic fork-join jobs in a multicomputer system. *IEEE Transactions on Parallel and Distributed Systems*, 4(10):1147–1164, 1993.

[38] S. Lavenberg. A perspective on queueing models of computer performance. *Performance Evaluation*, 10(1):53 – 76, 1989.

[39] E. L. Lawler. *Combinatorial optimization: Networks and matroids*. Holt, Rinehart and Winston, New York, 1976.

[40] E. K. Lee and R. H. Katz. An analytic performance model of disk arrays. *SIGMETRICS Perform. Eval. Rev.*, 21(1):98–109, 1993.

[41] H. Li and S. H. Xu. On the dependence structure and bounds of correlated parallel queues and their applications to synchronized stochastic systems. *Journal of Applied Probability*, 37(4):1020–1043, 2000.

[42] Y. Liu. Queueing network modeling of elementary mental processes. *Psychological Review*, 103(1):116–136, 1996.

[43] R. Luce and A. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14:95–116, 1949.

[44] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, 1979.

[45] J. Menon. Performance of RAID5 disk arrays with read and write caching. *Distributed and Parallel Databases*, 2(3):261–293, 1994.

[46] J. Menon and D. Mattson. Performance of disk arrays in transaction processing environments. In *Proceedings of the 12th International Conference on Distributed Computing Systems*, pages 302–309. IEEE, 1992.

[47] R. Nelson and A. N. Tantawi. Approximate analysis of fork/join synchronization in parallel queues. *IEEE Transactions on Computers*, 37(6):739–743, 1988.

[48] R. Nelson, D. Towsley, and A. Tantawi. Performance analysis of parallel processing systems. *IEEE Transactions on Software Engineering*, 14(4):532–540, 1988.

[49] V. Nguyen. Processing networks with parallel and sequential tasks: Heavy traffic analysis and brownian limits. *The Annals of Applied Probability*, 3(1):28–55, 1993.

[50] J. B. Orlin and R. K. Ahuja. New scaling algorithms for the assignment and minimum mean cycle problems. *Mathematical Programming*, 54:41–56, 1992.

[51] O. A. Prokopyev, H. Huang, and P. M. Pardalos. On complexity of unconstrained hyperbolic 0-1 programming problems. *Operations Research Letters*, 33(3):312–318, 2005.

[52] T. Radzik. Newton's method for fractional combinatorial optimization. In *Proceedings of 33rd Annual Symposium on Foundations of Computer Science*, pages 659–669, 1992.

[53] R. Sankaranarayana and L. Stewart. Complexity results for well-covered graphs. *Networks*, 22:247–262, 1992.

[54] M. Shigeno, Y. Saruwatari, and T. Matsui. An algorithm for fractional assignment problems. *Discrete Applied Mathematics*, 56:333–343, 1995.

[55] M. Tawarmalani, S. Ahmed, and N. V. Sahinidis. Global optimization of 0-1 hyperbolic programs. *Journal of Global Optimization*, 24:385–416, 2002.

[56] A. Thomasian and J. Menon. RAID5 performance with distributed sparing. *IEEE Transactions on Parallel and Distributed Systems*, 8(6):640–657, 1997.

[57] A. Thomasian and A. N. Tantawi. Approximate solutions for M/G/1 fork/join synchronization. In *Proceedings of the 26th Conference on Winter Simulation*, pages 361–368, 1994.

[58] D. Towsley, C. Rommel, and J. Stankovic. Analysis of fork-join program response times on multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 1(3):286–303, 1990.

[59] E. Varki. Mean value technique for closed fork-join networks. In *ACM SIGMETRICS Performance Evaluation Review*, volume 27, pages 103–112. ACM, 1999.

[60] E. Varki. Response time analysis of parallel computer and storage systems. *IEEE Transactions on Parallel and Distributed Systems*, 12(11):1146–1161, 2001.

[61] S. Varma and A. M. Makowski. Interpolation approximations for symmetric fork-join queues. *Performance Evaluation*, 20(13):245 – 265, 1994.

[62] R. Wiser, E. Lantz, M. Bolinger, and M. Hand. Recent developments in the levelized cost of energy from us wind power projects. `http://eetd.lbl.gov/ea/emp/reports/wind-energy-costs-2-2012.pdf`, 2012. Last accessed: October 2015.

[63] S. Wu, S. Jiang, B. C. Ooi, and K.-L. Tan. Distributed online aggregations. In *Proceedings of the VLDB Endowment*, volume 2, pages 443–454, 2009.

[64] T. Wu. A note on a global approach for general 0-1 fractional programming. *European Journal of Operational Research*, 101(1):220–223, 1997.