Exploiting Limited Customer Choice and Server Flexibility

Exploiting Limited Customer Choice and Server Flexibility

By

Yu-Tong HE, B.Eng.(Hons.), M.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree of
Doctor of Philosophy

McMaster University

DOCTOR OF PHILOSOPHY (2008) MCMASTER UNIVERSITY
(Software Engineering) Hamilton, Ontario

TITLE: Exploiting Limited Customer Choice and Server Flexibility

AUTHOR: Yu-Tong HE
B.Eng.(Hons.), Shanghai Jiao Tong University, China
M.Sc., McMaster University, Canada

SUPERVISOR: Dr. Douglas G. Down

NUMBER OF PAGES: xi, 159

Abstract

Flexible queuing systems arise in a variety of applications, such as service operations, computer/communication systems and manufacturing. In such a system, customer types vary in the flexibility of choosing servers; servers vary in the flexibility of which types of customers to serve. This thesis studies several resource allocation policies which address the concerns of limited customer choice and server flexibility. First, to accommodate different levels of flexibility, we propose the MinDrift affinity routing (MARO) policy and three variants: MARO$-2/k$, MARO$-$flex and MARO$-$tree. These policies are designed to maximize the system capacity by using the first moments of the inter-arrival times and the service times, at the same time they require only a small amount of state information in minimizing the delay in the system. Using diffusion limits for systems with Poisson arrival processes, we prove that MARO, MARO$-$flex and MARO$-$tree have the same heavy traffic optimality properties and the optimality is achieved independent of the flexibility levels. By providing their applications in distributed computing systems, we show that the MARO related policies (which require significantly less state information) outperform the MinDrift(Q) policy (which requires global state information), in heterogeneous server systems with either high or medium loads. Second, when no state information is available, we propose both the random routing policy which asymptotically minimizes the delay in the system by using the second moments of the service times, and the pooling strategy which further reduces the delay by combining appropriate parallel single-server queues into a number of multi-server queues. Overall, this thesis intends to provide insights on designing effective policies for allocating servers' times to serve multiple types of customers.

Acknowledgments

First of all, I shall express my sincere gratitude to Dr. Douglas G. Down, my supervisor, for offering his patient guidance and invaluable support. To him, I am indebted for the blend of challenges and freedom that allowed me to explore this research field.

I am grateful to Drs. George Karakostas, Terence Todd, Norm Archer and Otis Jennings for their careful review of this thesis and for sharing with me their uncommon wealth of insights.

My appreciation goes to Laurie LeBlanc for her help with administrative matters. Also, I shall say thank-you to my fellow students for adding colours to daily life.

I thank my parents, my sister and her family for their constant understanding and moral support; they keep calling me hurry up, from their time zone 12 hours ahead of mine.

Lastly, but not least, I should thank McMaster University for having supported this work through scholarships.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis studies resource allocation in stochastic service systems with multiple types of tasks and heterogeneous servers. The system can be referred to as a flexible queueing system, in which task types vary in the inter-arrival time distributions and in the flexibility of server choice for processing; servers vary in the service time distributions and in the flexibility of which types of tasks to process. Flexible queuing systems arise in a variety of applications, including service operations [32], computer systems [42] and manufacturing [16]. Resource allocation for such systems deals with how to allocate a server's time to process different types of tasks, in order to achieve certain performance goals (e.g., a task's mean sojourn time in the system is minimized).

In this chapter, we will describe the system models of interest, the issues for resource allocation and the methodology adopted in our study.

1.1 Motivation

1.1.1 System Models

For flexible queueing systems, there are two related queueing models: output-queued and input-queued systems. Both systems are equipped with independent parallel queues and a number of servers. Each queue might be associated with multiple servers. Tasks arrive from outside of the system and wait in one of the queues until served. They require a single service and leave the system

upon completion.

Figure 1.1 shows an output-queued system. In this model, each task must be routed (by a dispatcher) to one of the servers immediately upon arrival. After being routed, the task stays in the server's queue until it is processed. Such models can be found in applications such as manufacturing systems [64], wireless networks [63], medical services and distributed computing systems (or similarly, multi-location call centres [4]). For example, a wireless application may have data packets to be transmitted to multiple destinations. These data packets constitute multiple types of tasks and are dispatched to a set of transmitters whose transmission rates depend on the channel qualities between different transmitters and destinations.

Figure 1.2 shows an input-queued system, where tasks of the same type are placed in a dedicated input queue without being pre-assigned to any particular server. When a task is at the head of the queue, it is taken by one of the servers which is ready for processing. A scheduling policy is required to specify, at each time a server completes the processing of a task, which task type that server will process next. This involves both routing decisions (to direct which task types to which servers) and sequencing decisions (to determine the order in which servers will process tasks from their dedicated queues). This model arises in applications like cross-bar switches [62] and single-location call centres [8, 69].

In this thesis, we study the resource allocation problem for output-queued systems and demonstrate its applications in distributed computing systems and medical services.

1.1.2 Issues in Resource Allocation

One of the most commonly managed qualities of a queueing system is the accessibility of servers, i.e., "How long did customers have to wait to receive service". To reduce the delay in an output-queued system, resource allocation can take three measures: (1) the routing policy that decides which types of tasks are routed to which queues; (2) the pooling strategy that combines different subsets of single-server queues into multi-server queues; and (3) the

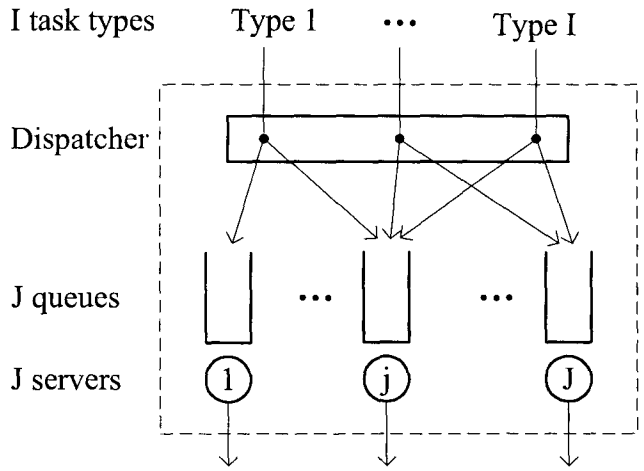

Figure 1.1: An output-queued system

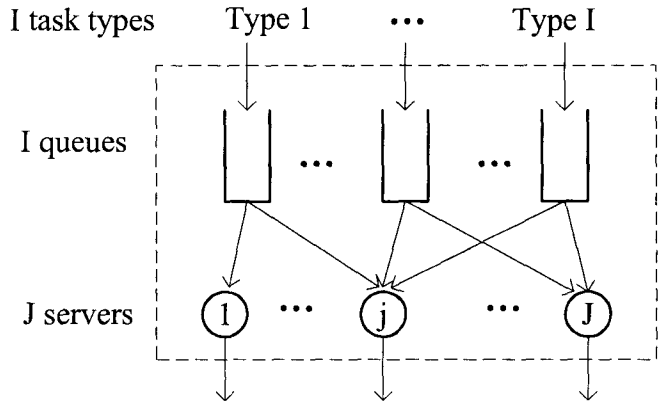

Figure 1.2: An input-queued system

local scheduling policy that determines the order in which the waiting tasks are processed at each server. Different policies require various amount of system information, which impacts the complexity of implementation. For example, designing a new scheduling policy in a computer server system requires the control of a server's kernel, which sometimes is hard to obtain. In this thesis, we focus on the routing policy and the pooling structure, assuming that the local scheduling is non-preemptive and all task types have the same priority.

Information Needed

The information needed for making routing decisions is classified into two major categories. One is the *state information*, which includes the actual numbers of tasks waiting in queues and/or in process, the expected completion times of a new arrival at different queues, and the actual remaining processing time at each server. The other is the *primitives*, which are distributions of task inter-arrival times and processing times, or individual task sizes. Within each category, differing amounts of information may be required for a routing policy. For example, a dynamic routing policy requires current state information (and sometimes the past routing history), which is clearly more complex than a static routing policy that requires no state information.

State information can enable a dynamic policy to achieve better performance than a static policy, but it also affects scalability. There are several problems with requiring up-to-date state information from a large number of servers. Firstly, the complexity of implementation increases, which can make the policy infeasible. For example, an unreasonably long time might be required for collecting and processing the state information before a routing decision can be made. Secondly, even if the policy is feasible, a large amount of state information increases the system load by adding overhead in routing individual tasks, which may compromise the gains in performance. Thirdly, the supplied state information may be out of date when a routing decision is made. As observed in [50], performance degradation resulting from outdated information is a major limitation of policies which require global state information. Therefore, tradeoffs between the performance gain and the amount of

required state information should be evaluated. In this thesis, we propose a series of routing policies to address these problems by requiring dramatically less state information while at the same time achieving competitive performance.

Regarding the information of primitives, our proposed routing policies require no a priori knowledge of each task's size, but the first (and in some cases the second) moment(s) of the task inter-arrival times and processing times. For the case where individual task sizes are known, a size-interval strategy [21] is proved to be the optimal static routing policy for a homogeneous system of first-come-first-serve queues. However, in the situation where task size and processing time have very little correlation (e.g., as discussed by Lu et al. in [45]), even if the task size is known to be fixed, it is more effective to know the processing time distributions at the servers.

Server and Customer Flexibility

In many applications, servers are partitioned into specialized and cross-trained groups. For example, in emergency medical services, there are both basic and advanced life support units [40]; in call centres, customer service representatives are unilingual or multilingual [25]; and in web applications, servers can be configured as specialized or general servers [13]. These configurations of servers are not only to cope with the various demands from customers, but in the latter case of web applications also to improve scalability of the system.

To improve service quality, we are faced with two design questions: (1) how much flexibility should each server have and (2) how much routing flexibility should be provided to each customer type. Consequently, the system performance can be compared in two scenarios: dedicated systems versus flexible systems, where the difference in performance is affected by both the routing policy and the pooling strategy.

In a dedicated system, a customer has no choice in which queue to receive service. So no state information is needed for making routing decisions. An extreme case is that each customer type can be routed to only one server and each server can only serve one customer type, where no pooling is allowed. In such a case, each server is configured for exactly one kind of task, which in

turn should reduce the cost of implementing and maintaining high flexibility for the servers.

In a flexible system, a customer has the flexibility of choosing a subset of queues to join (a strict subset may be due to locality constraints or personal preference). Servers are partitioned into a number of subsets, where in each subset, the servers are capable of serving the same types of customers that are waiting in a single queue. When the number of servers in any subset is greater than one, a pooling structure is formed. Pooling identical servers result in performance improvement (e.g., one $M/M/2$ queue versus two $M/M/1$ queues), however, it is not always superior in a heterogeneous system. Buzacott [15] shows that when the service times of different customer types are not identical in distribution, pooling can lead to longer queueing delays. On the other hand, Andradóttir et al. [7] shows that a static routing policy which maximizes the system capacity does require a subset of servers capable of serving multiple customer types. Therefore, there is a tradeoff between the system performance and the cost of maintaining highly flexible servers.

1.1.3 Heavy Traffic Analysis

To obtain insight into the design of resource allocation policies for output-queued systems, we employ the method of diffusion limit analysis under heavy traffic conditions [68]. Intuitively speaking, servers in a heavy traffic system are operating with nearly full utilization.

In this thesis, heavy traffic limits for a queueing model are obtained by considering a sequence of queueing models, where the input load is allowed to increase toward the maximum possible, while maintaining stability. Under heavy traffic conditions, the steady state performance measures (e.g., the queue lengths) typically grow unboundedly. However, with appropriate scaling of both time and space, one can construct for the entire queue length process a stochastic-process limit. For example in the standard case, such a limit is a Markov process with continuous sample paths, also called a diffusion process (due to the historic connection with mathematics of physical diffusion).

The heavy traffic analysis method has several attractive features: (1) it

produces simple heavy-traffic limits which identify the key elements affecting the performance of a complicated system; (2) it yields asymptotically exact behaviour of the system in steady state, which is otherwise analytically intractable, given maximum possible input load; (3) it reduces the dimension of the state space in the system control problem, providing a form of state-space collapse which makes the problem easier to analyze.

Our objective of resource allocation is to identify routing policies and pooling strategies that are asymptotically optimal in heavy traffic, i.e., minimizing an arrival's mean sojourn time in the system, or equivalently by Little's Law, minimizing the mean total number of tasks in the system. We would like to achieve this minimum using the smallest amount of server flexibility, while accommodating different levels of customer flexibility.

To determine routing policies in Markovian queueing systems, dynamic programming (DP) [57] is a standard approach. However, for a system with many servers and many customer types, the state space may be uncountably infinite, which makes it very difficult to derive structural properties of effective policies. The application of standard DP techniques to numerically find optimal controls also becomes intractable.

Due to the inherent complexity of the systems under consideration in this thesis, heavy traffic analysis would be appropriate, not only because of its engineering significance mentioned above, but also because in general, the case of interest is when the system is heavily loaded (both as being worst case and economically).

1.2 Related Work

In the literature on heavy traffic analysis, two asymptotic regimes are discussed in the design and control of multi-server queueing systems.

The first is the so-called "finite-server" heavy traffic regime (first established by Kingman [41], also referred to as the conventional heavy traffic regime). In this regime, the system has a fixed finite number (m) of servers and the system utilization approaches one, the critical value for stability. This regime corresponds to the operating environment where server costs dominate

the cost of customer delay and where almost all customers may encounter delays [25].

Iglehart and Whitt [39] prove that in heavy traffic, the scaled queue-length processes are asymptotically equivalent in the following two systems: a $G/G/1$ queue with service rate m and a $G/G/m$ queue (each of the m servers with rate 1), both having identical arrival processes. That is, for the multi-server model with homogeneous servers and the FCFS discipline, the m servers act as a "single super server". This is also true for the system with the join-the-shortest-queue (JSQ) rule, where the multi-dimensional queue length process collapses to one dimension in heavy traffic [58]. This state space collapse enables the sojourn time processes in different queues to be expressed in terms of the limit of the one-dimensional total queue-length process.

The second regime is the so-called Halfin-Whitt regime [31]. Here the arrival rate and the number of servers are scaled up (to infinity) so that the desired traffic load is obtained and the steady state probability that a customer gets delayed converges to a limit $0 < \alpha < 1$. This regime is more likely to be used in describing the dynamics of a call centre where the staffing level grows in response to the high arrival rate of incoming calls. However this is not the case on which we are focusing, where the number of servers in the system is not necessarily large and the service rate of each server is not fixed. Moreover, to analyze a system in this regime requires one to solve a non-linear partial differential equation whose dimension equals the number of customer types, and is therefore not practical to derive implementable control policies [6, 8].

Armony and Maglaras [5] perform an asymptotic analysis in the Halfin-Whitt regime of an $M/M/m$ system, which models call centres with a call-back option for customers. The model is restricted to one with 2 customer classes and a single server group, however, it is still interesting to see that introducing this customer choice also benefits those customers who do not make this choice (modelled as one of the two customer classes): the asymptotic approximation of their expected waiting time is reduced (by more than 50%), when all the customers have static estimates of the steady-state waiting time.

In a system with multiple types of customers served by groups of specialized and cross-trained servers, *skills-based routing* [25] becomes a neces-

sity. Garnett and Mandelbaum [26] define some canonical designs of network topologies, which represent building blocks for more complex parallel queueing systems. For example, in a "W" design, two pools of agents cater to three types of customers: Pool 1 serves Type 1 and 2; Pool 2 serves Type 2 and 3. In an "X" design, two pools of servers have full flexibility so that two types of customers can be routed to either pool. They use simulation to demonstrate how various routing policies can yield a dramatic difference in system performance.

Harrison and López [35] give a skills-based routing model that resembles the reality of a call centre in the efficiency-driven regime. They identify a complete resource pooling (CRP) condition in which the set of parallel servers act as a single (pooled) super-server, however, the "X" design mentioned above does not satisfy this condition. Mandelbaum and Stolyar [48] prove that, for input-queued systems which satisfy the CRP condition, a simple generalized $c\mu$ (G$c\mu$) rule is asymptotically optimal in heavy traffic. Each idle server chooses to serve the longest-waiting customer, who incurs a convex increasing waiting cost. So the G$c\mu$ rule performs well for systems that are efficiency driven. The rule does not depend on arrival rates, but requires continuous re-calculation of the state-dependent waiting cost of each present customer. Stolyar proves in [63] that for output-queued systems under the CRP condition, the MinDrift routing rule (in conjunction with arbitrary work-conserving disciplines at the servers) has asymptotic optimality properties analogous to those of the G$c\mu$ rule for input-queued systems. The difference is that waiting cost is calculated once at the arrival time of each customer. Teh and Ward [64] prove for a two-queue system the asymptotic optimality of a threshold routing policy, under linear holding costs. Under this policy, arrivals are routed to the fast-server queue 1 unless the other slow-server queue 2 is below the threshold. However, this threshold policy is outperformed by the shortest-expected-delay-routing policy. The reason (conjectured by the authors) is because the threshold policy keeps track of less state information (only one queue length instead of two).

For routing policies in flexible queueing systems, we are also interested in the optimality of JSQ schemes where not all arrivals join the shortest queues. It is known that JSQ routing (with FCFS service discipline) minimizes the

mean waiting time for exponentially distributed interarrival and service times, under knowledge of full state information [70]. Whitt [66], however, provided several examples in which JSQ routing is not optimal when service times have large variance. Foley and McDonald [22] analyze the asymptotic behaviour of JSQ in a "W" design system, where the service rates are server-dependent. They determine the limiting distributions of the queue lengths when the total number of customers in the system grows high, and suggest that to balance the load, a potential least-costly solution is to route a small portion of the arrivals to the shorter queue, while having most of the customers remain dedicated to their servers.

The literature on pooling usually studies the comparison between two extreme cases: a dedicated system versus a fully pooled system. In the case of partial pooling, two issues are involved: 1) how many dedicated servers to pool for cross-training and 2) which servers to pool.

Smith and Whitt [61] show that a pooled system can be made arbitrarily worse than a dedicated one if there are rare customers with long service times. It is also shown that in systems with homogenous service and demand distributions, a pooled system always outperforms a dedicated one.

Mandelbuam and Reiman [47] reduce the pooling of a Jackson network into an $M/PH/1$ queue. A stream of incoming customers change their types by demanding different service times for different tasks. An efficiency index is used to show that pooling always helps in light traffic, while its effects can go either way in heavy traffic. Pooling a parallel structure can sometimes hurt unboundedly if the service rates at each queue are sufficiently different.

Gurumurthi and Benjaafar [30] use a continuous time Markov chain to model flexible queueing systems, where an arbitrary number of Poisson arrivals have the flexibility of being routed to more than one server based on a priority scheme, while the service times are exponentially distributed with means which depend on the servers. Their characterization of the probability distribution of system states and the transition probabilities between these states is claimed to offer the opportunity to formulate optimal control problems (e.g., on capacity allocation) in the framework of a Markov decision process. One of the related studies is by Shumsky [60], where performance measures for an "N" design

system are generated by an approximation procedure that decomposes the state space of the Markov chain.

Tekin *et al.* [65] use Kingman's Law of Congestion (see [67]) to approximate both the average delay W_i in an $M/G/m_i$ queue (each queue dedicated to type i customers, $i = 1, ..., N$) and the average delay W_K in one pooled department with K $M/G/m$ queues, $K < N$. It is shown that to minimize W_K, the selection of these K queues depends on the system parameters, such as arrival rates, service times and the number of servers m_i of each queue. A sufficient condition is provided for pooling to be advantageous in systems with uniform utilization and service time variability. This condition implies that the queues to be pooled should have mean service times close to each other, high service time SCV's (squared coefficient of variation) and small m_i.

Gans *et al.* [25] study the pooling of geographically dispersed call centres, the base case of which is the $M/M/m$ (Erlang C) model. The pooling may be obtained either physically by combining two or more centres into a larger one, or virtually through the use of networking technology that connects calls to various sites. They show that under efficiency-driven staffing, the scaled expected waiting time before being served (or ASA, average speed of answer in the call-centre context) is unchanged with respect to the base case. Here all servers in a pool can handle the same set of customer types.

1.3 Contributions

The main contributions of this thesis include:

1. We propose a MinDrift Affinity Routing (MARO) policy for output-queued systems with heterogeneous servers to process multiple types of tasks. Given the first moments of both the task inter-arrival times and the task processing times, this policy is designed to both maximize the capacity of the system and minimize the delay in the system by using a subset of global state information.

2. We propose three variants of the MARO policy, namely MARO$-2/k$, MARO$-$flex and MARO$-$tree, to accommodate different levels of flexi-

bility with which tasks acquire limited amounts of state information for routing. Using diffusion limits for systems with Poisson arrival processes, we prove that MARO−flex and MARO−tree have the same heavy traffic optimality properties as does MinDrift(Q) and the optimality is achieved independent of the levels of flexibility. For the special cases of systems with a single task type, MARO−flex and MARO−tree approach the lower bound of achievable performance.

3. We propose resource allocation policies for output-queued systems with homogeneous servers and Poisson arrival processes, where no state information is available. The proposed policies consist of two parts: one is the random routing policy which asymptotically minimizes the delay in the system under heavy traffic, given the first and second moments of the task processing times; the other is the pooling strategy which combines the parallel single-server queues into a number of multi-server queues, in order to further reduce the delay in the system.

4. We make extensions of the optimal random routing policies for two situations: one is heterogeneous server systems with Poisson arrival processes, which operate under medium load; the other is heterogeneous systems in heavy traffic with generic arrival processes.

5. We provide applications of the MARO related policies in heterogenous computing systems and in medical services (or service operations in general). Guidelines for choosing the routing policies in system design are also provided.

1.4 Thesis Outline

The remainder of this thesis consists of the following chapters and appendices.

Chapter 2 introduces the related mathematical background. For completeness, theorems later used in the thesis are included.

Chapter 3 presents the proposed MARO policies and the resource allocation LP on which the policies are based. The heavy traffic optimality

properties of these policies are proved and the discounted amount of required state information is compared for three variants of MARO. Simplified results are given for homogeneous/heterogeneous systems with a single type of tasks.

Chapter 4 discusses the resource allocation policies which require no state information. To derive the optimal random routing matrices, three nonlinear programming (NLP) problems are formulated for different system configurations, respectively. An alternative way of solving the NLPs using geometric programming is discussed in Appendix A. For homogenous systems in heavy traffic, a procedure for choosing pooling strategies to further reduce the system delay is proposed. A heuristic which produces the routing matrix for a higher degree of pooling is shown in Appendix B as a complement to the NLP method.

Chapter 5 demonstrates the applications of the MARO variants in Grid systems (which are generalized distributed computer systems). Issues of applying two related policies to reduce hospital waiting times are discussed. Implications from these applications are summarized for system design.

Chapter 6 includes some concluding remarks and suggestions for future work.

Chapter 2

Mathematical Background

In this chapter, we introduce required definitions of stochastic process limits and collect some existing results in heavy traffic analysis that will be used in our study. Section 2.1 presents the concepts of weak convergence in the Skorohod space with uniform topology, and reflected Brownian motion (RBM), which are together used to establish diffusion limits for stochastic processes using the functional central limit theorem (FCLT). Section 2.2 includes heavy traffic theorems for a single queue (with either one server or multiple servers), where the stochastic processes of interest are the one-dimensional queue length and waiting time processes. In Section 2.3, we state the corresponding theorems for output-queued systems and the complete resource pooling (CRP) condition, under which a parallel queue system effectively forms a single pool of processing capacity and the state space of the system information collapses into one dimension, making the system much easier to analyze.

2.1 Stochastic Process Limits

Materials in this section follow Chapters 3 and 4 in [68] and Chapter 5 in [19].

2.1.1 Weak Convergence

Let the metric space (S, m) be endowed with the Borel σ-field $\mathcal{B}(S)$ and X be a mapping from a probability space $(\Omega, \mathcal{F}, \mathcal{P})$ to $(S, \mathcal{B}(S))$. If X is measurable,

i.e, if $\{\omega \in \Omega : X(\omega) \in A\} \in \mathcal{F}$ for all $A \in \mathcal{B}(S)$, we call it a random element of $(S, \mathcal{B}(S))$. If S is the K-dimensional Euclidean space $\mathbb{R}^K$, X is called a K-dimensional random vector. If S is a space of K-dimensional real-valued functions which are defined on the subinterval $[0, T]$ of the real line and are right-continuous with left limits, X is called a K-dimensional stochastic process. The corresponding function space is denoted as $\mathcal{D}^K := \mathcal{D}([0, T], \mathbb{R}^K)$, or simply $\mathcal{D}$. For the subspace $\mathcal{C}$ of continuous functions, the reference metric $m(s_1, s_2)$ $(s_1, s_2 \in S)$ is the uniform metric $||s_1 - s_2||$, defined in terms of the uniform norm

$$||x|| := \sup_{0 \le t \le T} \left\{ \max_{1 \le k \le K} |x_k(t)| \right\},$$

for all $x \in S$, where $x := \{x(t) : 0 \le t \le T\}$ and $x(t) := [x_1(t), \ldots, x_K(t)] \in \mathbb{R}^K$. In this case, the space $\mathcal{D}$ is a (standard) Skorohod space with uniform topology, which is sufficient for our study, because all of the limit processes considered in this thesis have continuous sample paths.

The distribution of X is the image probability measure P induced by X on $(S, \mathcal{B}(S))$, denoted as

$$P(A) := \mathcal{P}(X \in A) := \mathcal{P}(\{\omega \in \Omega : X(\omega) \in A\}), \quad A \in \mathcal{B}(S).$$

Let X be a stochastic process and $\{X_n : n \ge 1\}$ be a sequence of stochastic processes, all defined on the probability space $(\Omega, \mathcal{F}, \mathcal{P})$. Let P and P_n be the distributions of X and X_n, respectively. We say that P_n converges weakly to P if for every bounded and continuous function f on $\mathcal{D}$,

$$\lim_{n \to \infty} \int_{\mathcal{D}} f dP_n = \int_{\mathcal{D}} f dP.$$

In other words, X_n converges weakly to X (or X_n converges to X in distribution), denoted by $X_n \xrightarrow{w} X$, if and only if

$$\lim_{n \to \infty} E[f(X_n)] = E[f(X)],$$

for every f.

Given the established stochastic process limits, new stochastic process limits may be obtained using the following theorems.

Let $\{X_n : n \geq 1\}$ and $\{Y_n : n \geq 1\}$ be two sequences of random elements which are defined on a common domain in a separable metric space (S, m). (By separable, we mean that there is a countable subset $S_0 \subseteq S$ such that $\forall s_1 \in S$, $\forall \epsilon > 0$, $\exists s_2 \in S_0$, $m(s_1, s_2) < \epsilon$.)

Theorem 2.1.1 ([68], Theorem 11.4.7, convergence together theorem). *If* $X_n \xrightarrow{w} X$ *in* (S, m) *and* $m(X_n, Y_n) \xrightarrow{w} 0$ *in* $\mathbb{R}$, *then*

$$(X_n, Y_n) \xrightarrow{w} (X, X) \quad in \quad (S, m) \times (S, m).$$

The converse to Theorem 2.1.1 yields not only two marginal limits $X_n \xrightarrow{w} X$ and $Y_n \xrightarrow{w} X$, but also asymptotic equivalence of X_n and Y_n.

Let g be a Borel measurable function mapping a separable metric space (S, m) into another separable metric space (S', m'). Let D_g be the set of discontinuity points of g, i.e., D_g is the subset of $s \in S$ such that there exists a sequence $\{s_n : n \geq 1\}$ in S with $\lim_{n\to\infty} m(s_n, s) = 0$ and $\lim_{n\to\infty} m'(g(s_n), g(s)) \neq 0$.

Theorem 2.1.2 ([68], Theorem 3.4.3, continuous mapping theorem). *If* $X_n \xrightarrow{w} X$ *in* (S, m) *and* $g : (S, m) \to (S', m')$ *is measurable with* $\mathcal{P}(D_g) = 0$, *then* $g(X_n) \xrightarrow{w} g(X)$.

Let $\mathcal{D}_0$ be the subset of all $x \in \mathcal{D}^K$ with $x_k(0) \geq 0$ for all $1 \leq k \leq K$. Let $\mathcal{D}_\uparrow$ be the subset of functions in $\mathcal{D}_0$ that are nondecreasing. Let $\mathcal{C}_0$, $\mathcal{C}_\uparrow$ be the corresponding subsets of $\mathcal{C}^K$; i.e., $\mathcal{C}_0 = \mathcal{C} \cap \mathcal{D}_0$, $\mathcal{C}_\uparrow = \mathcal{C} \cap \mathcal{D}_\uparrow$.

Lemma 2.1.3 ([68], Theorem 13.2.1). *The composition map* $\delta \circ x(t) := \delta(x(t))$ *from* $\mathcal{D}_\uparrow^1 \times \mathcal{D}^K$ *to* $\mathcal{D}^K$ *is measurable and continuous at* $(\delta, x) \in \mathcal{C}_\uparrow^1 \times \mathcal{C}^K$.

Suppose two random elements $X \in \mathcal{D}^K$ and $\Delta \in \mathcal{D}_\uparrow^1$ are defined on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$. Each element of the two sequences of random elements $\{X_n : n \geq 1\}$ and $\{\Delta_n : n \geq 1\}$ is defined on a probability space $(\Omega_n, \mathcal{F}_n, \mathcal{P}_n)$.

Theorem 2.1.4 ([12], (17.9), random time change theorem). *If* $(X_n, \Delta_n) \xrightarrow{w} (X, \Delta)$ *in* $\mathcal{D}^K$ *and* $\mathcal{P}(X \in \mathcal{C}^K) = \mathcal{P}(\Delta \in \mathcal{C}_\uparrow^1) = 1$, *then* $\Delta_n \circ X_n \xrightarrow{w} \Delta \circ X$ *in* $\mathcal{D}^K$.

2.1.2 Brownian Motion and Donsker's Theorem

The fundamental stochastic process limit is the convergence of a sequence of scaled random walks to Brownian motion (BM) in the function space $\mathcal{D}$. This is provided by Donsker's Theorem. Using a continuous mapping approach, new stochastic process limits can be established for queueing models.

A process $B_0 = \{B_0(t) : t \geq 0\}$ is a standard Brownian motion, or Wiener process, if it (1) has continuous sample paths with $B_0(0) = 0$, (2) has stationary and independent increments $B_0(t_n) - B_0(t_{n-1})$, $n \geq 1$ and (3) when evaluated at time t, is normally distributed with mean 0 and variance t, denoted as $B_0(t) \sim N(0, t)$. As random variables, the increments are independent for any $n < \infty$ and $0 \leq t_0 < t_1 < \ldots < t_n$. They are stationary if the distribution of $B_0(t + h) - B_0(t)$ depends only on $h > 0$.

A process $B = \{B(t) : t \geq 0\}$ defined by $B(t) = B(0) + \theta t + \sigma B_0(t)$ is called a Brownian motion with drift θ and variance σ^2 starting at $B(0)$, denoted as $BM(\theta, \sigma^2)$. It follows that $B(t+h) - B(t) \sim N(h\theta, h\sigma^2)$. Brownian motion is a diffusion process because it satisfies the Markov property ([34], page 5).

Theorem 2.1.5 ([19], Theorem 5.7, Donsker's theorem). *Let $\{x_i : i \geq 1\}$ be a sequence of independent and identically distributed (i.i.d.) random variables. Assume that $E[x_1] = \mu^{-1} < \infty$ and $Var[x_1] = \beta^2 < \infty$. For each $n \geq 1$, define a scaled random walk process $\hat{X}^{(n)} = \{\hat{X}^{(n)}(t), t \geq 0\}$ with the centred partial sums*

$$\hat{X}^{(n)}(t) = \frac{1}{\beta\sqrt{n}} \left(\sum_{i=1}^{\lfloor nt \rfloor} x_i - nt\mu^{-1} \right),$$

where the summation is understood to be zero when $nt < 1$. Then as $n \to \infty$, $\hat{X}^{(n)} \xrightarrow{w} B_0 = BM(0, 1)$.

Donsker's theorem is called a functional central limit theorem (FCLT), since it is a generalization of the classic CLT. This can be seen by fixing the time t for each process $\hat{X}^{(n)}$.

Let $X = \{X(k) : k \geq 1\}$ be the unscaled random walk with partial sums $X(k) = \sum_{i=1}^{k} x_i$, where each x_i is nonnegative. Define its associated

counting process $Y = \{Y(t), t \geq 0\}$ by

$$Y(t) = \sup\{k : X(k) \leq t\}.$$

Since each x_i is an i.i.d. random variable, $Y(t)$ is a renewal process. For each $n \geq 1$, define the scaled process $\hat{Y}^{(n)} = \{\hat{Y}^{(n)}(t), t \geq 0\}$ by

$$\hat{Y}^{(n)}(t) = \frac{1}{\sqrt{n}} \left(Y(nt) - nt\mu\right).$$

Theorem 2.1.6 ([19], Theorem 5.11). *If the conditions in Theorem 2.1.5 hold for the sequence* $\{x_i : i \geq 1\}$, *then* $\hat{Y}^{(n)} \xrightarrow{w} BM(0, \mu^3\beta^2)$, *as* $n \to \infty$.

2.1.3 Reflected Brownian Motion

Define a one-dimensional, one-sided reflection mapping $\Delta(\cdot)$ as

$$\Delta(x(t)) := x(t) + \sup_{0 \leq s \leq t} [x(s)]^-,$$

where $[x]^- := \max(-x, 0) := [-x]^+$ is the negative part of x. The mapping $\Delta(\cdot)$ is Lipschitz continuous on $\mathcal{D}_0$ and its effect is shown in Figure 2.1. The origin with respect to $z(t) = \Delta(x(t))$ appears to be the thick dashed line, where $z(t)$ equals $x(t)$ up until the first time t at which $x(t) = 0$ and thereafter $z(t)$ equals the amount by which $x(t)$ exceeds the minimum value of x over $[0, t]$. Therefore, such a mapping is also called a one-sided *regulator* with lower barrier at zero. This transform can be used to model queueing processes in a system with unlimited waiting space [68].

If $X = BM(\theta, \sigma^2)$ and $Z(t) = \Delta(X(t))$, a process $Z = \{Z(t) : t \geq 0\}$ is a called a reflected Brownian motion (RBM) with drift θ and variance σ^2, denoted as $RBM(\theta, \sigma^2)$. Unlike Brownian motion, RBM does not have independent increments, but it is still a Markov process. The process $RBM(0, 1)$ has the same distribution as the process $\{|B_0(t)| : t \geq 0\}$.

Theorem 2.1.7 ([19], Theorem 6.2). *If* $Z = RBM(\theta, \sigma^2)$ *is a one-dimensional RBM, then the process* Z *has a stationary distribution if and only if* $\theta < 0$, *in which case the stationary distribution is exponential with mean* $\varphi = \sigma^2/(2|\theta|)$.

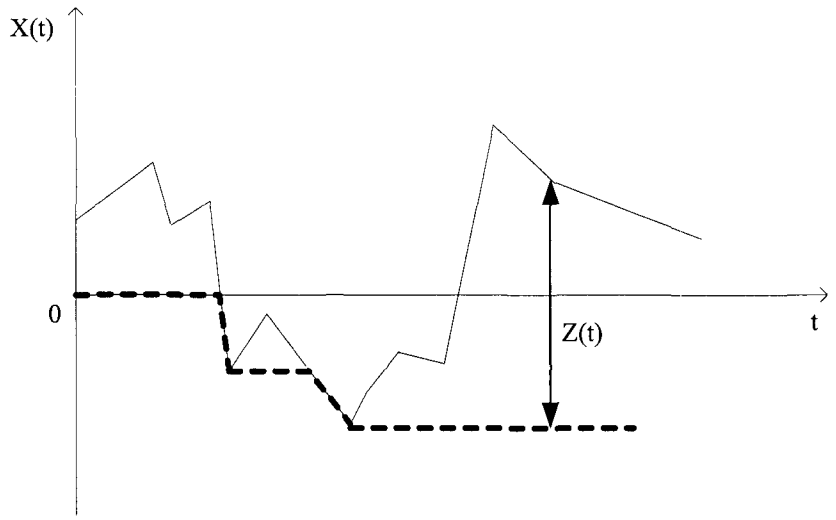

Figure 2.1: One-sided reflection mapping

2.2 Heavy Traffic Results for A Single Queue

In this section, we follow Reiman [58] to state several heavy traffic limit theorems for a single queue. First, the required probability space and the assumptions are described for a single queue system with I types of arrivals and J servers. Next, for the simple case of $I = J = 1$, diffusion limits are established for the queue length process and the sojourn time processes. Finally, limit theorems for a multi-server queue with I superimposed renewal arrival processes are introduced.

To identify the stochastic process limits for a queueing system, a sequence of queueing systems is considered. The motivation is as follows. The queue length process (or the sojourn time process) has a stationary distribution only when the traffic intensity ρ is strictly less than one. However, as will be shown, the diffusion limit of the queue length process is zero when $\rho < 1$. Therefore, the queueing system we are interested in is assumed to be an element in a sequence of systems whose traffic intensities approach one. Once the limit is obtained, it can be used to identify key elements that affect the performance of the system.

2.2.1 The Assumptions

Consider a queue with unlimited waiting space. Customers arrive one at a time and are served in the order of their arrival. On a probability space $(\Omega, \mathcal{F}, \mathcal{P})$, we define mutually independent sequences of nonnegative i.i.d. random variables, $\{u_{i,m} : m \geq 1\}$, $\{v_{j,\ell} : \ell \geq 1\}$, for all $1 \leq i \leq I$ and $1 \leq j \leq J$. Let $u_{i,m}$ be the inter-arrival time between the $(m-1)$-th and m-th customers of type i ($u_{i,1}$ being the arrival epoch of the first customer). Let $v_{j,\ell}$ be the ℓ-th service time at server j. We assume that both have finite means and finite variances:

$$E[u_{i,m}] = \lambda_i^{-1}, \qquad Var[u_{i,m}] = \alpha_i^2,$$
$$E[v_{j,\ell}] = \mu_j^{-1}, \qquad Var[v_{j,\ell}] = \beta_j^2.$$

Define random walks $U_i = \{U_i(k) : k \geq 1\}$ and $V_j = \{V_j(d) : d \geq 1\}$ with the partial sums

$$U_i(k) = \sum_{m=1}^{k} u_{i,m}, \quad V_j(d) = \sum_{\ell=1}^{d} v_{j,\ell}.$$

The associated counting processes are $A_i = \{A_i(t), t \geq 0\}$ and $S_j = \{S_j(t), t \geq 0\}$, with

$$A_i(t) = \sup\{k : U_i(k) \leq t\}, \quad S_j(t) = \sup\{d : V_j(d) \leq t\}.$$

For each type i, $U_i(k)$ is the arrival time of the k-th customer and $A_i(t)$ gives the number of customers that arrive during the time interval $[0, t]$. $S_j(t)$ is the number of customers that complete service at server j and depart the system, provided that the server is busy all of the time. We refer to $A(t) = \sum_{i=1}^{I} A_i(t)$ and $S(t) = \sum_{j=1}^{J} S_j(t)$ as the (superimposed) arrival process and the (potential) service process, respectively.

Now consider a sequence of queues as defined above, indexed by n. For the n-th queue, the random variables are defined on a probability space $(\Omega^{(n)}, \mathcal{F}^{(n)}, \mathcal{P}^{(n)})$. Let $(\lambda_i^{-1})^{(n)}$ and $(\alpha_i^2)^{(n)}$ be the mean and variance, respectively, for the inter-arrival times; $(\mu_j^{-1})^{(n)}$ and $(\beta_j^2)^{(n)}$ are the mean and variance, respectively, for the service times. We assume that for all $1 \leq i \leq I$ and

$1 \leq j \leq J$,

$$\lim_{n\to\infty} \lambda_i^{(n)} = \lambda_i, \qquad \lim_{n\to\infty} (\alpha_i^2)^{(n)} = \alpha_i^2, \tag{2.1}$$

$$\lim_{n\to\infty} \mu_j^{(n)} = \mu_j, \qquad \lim_{n\to\infty} (\beta_j^2)^{(n)} = \beta_j^2, \tag{2.2}$$

and

$$\sup_{n\geq 1} E\left[\left(u_{i,1}^{2+\epsilon}\right)^{(n)}\right] < \infty, \quad \sup_{n\geq 1} E\left[\left(v_{j,1}^{2+\epsilon}\right)^{(n)}\right] < \infty \tag{2.3}$$

for some $\epsilon > 0$. Condition (2.3) implies that both the inter-arrival times and the service times have finite means and variances. Moreover, under this condition, the functional central limit theorem still holds even if the time sequence is formed by random variables that are independent but not identically distributed (cf. the Lindeberg condition in the central limit theorem) [58]. They are used throughout the thesis for the same reason.

In addition, it is assumed that

$$\lim_{n\to\infty} \sqrt{n}\Big(\sum_{i=1}^{I} \lambda_i^{(n)} - \sum_{j=1}^{J} \mu_j^{(n)}\Big) \;=\; c < \infty \tag{2.4}$$

for some finite constant c. Let $\lambda = \sum_{i=1}^{I} \lambda_i$, $\mu = \sum_{j=1}^{J} \mu_j$ and $\rho = \lambda/\mu$. It is known from (2.4) that $c = -\infty$ corresponds to $\rho < 1$ and $c > -\infty$ corresponds to $\rho = 1$.

2.2.2 A Single-Server Queue

Let $I = J = 1$. Define the potential workload process

$$X(t) = V(A(t)) - t.$$

By workload, we mean the amount of time to empty the system, if there are no more arrivals. The actual workload process is

$$Z(t) = \Delta(X(t)) = X(t) + \sup_{0\leq s\leq t} [X(s)]^{-}. \tag{2.5}$$

So, the k-th arrival at time T_k has a sojourn time of

$$W(k) = Z(T_k). \tag{2.6}$$

Since the cumulative busy time in $[0,t]$ at the server is

$$B(t) = t - \sup_{0\le s\le t}[X(s)]^-,$$

and $S(B(t))$ counts the number of departures in $[0,t]$, the queue length process is

$$Q(t) = A(t) - S(B(t)). \tag{2.7}$$

Define the scaled processes for (2.5)–(2.7) as

$$\hat{Z}^{(n)}(t) = \frac{1}{\sqrt{n}} Z^{(n)}(nt), \tag{2.8}$$

$$\hat{W}^{(n)}(t) = \frac{1}{\sqrt{n}} W^{(n)}(\lfloor \lambda nt \rfloor), \tag{2.9}$$

$$\hat{Q}^{(n)}(t) = \frac{1}{\sqrt{n}} Q^{(n)}(nt). \tag{2.10}$$

Theorem 2.2.1 ([58], Theorem 3.1). *If* (2.1)–(2.4) *hold with* $c > -\infty$, $\hat{Z}^{(n)}(t) \xrightarrow{w} \hat{Z} = RBM(c/\mu, \lambda(\alpha^2+\beta^2))$ *in* $\mathcal{D}$.

Theorem 2.2.2 ([58], Theorem 3.2). *If* (2.1)–(2.4) *hold with* $c = -\infty$, $\hat{Z}^{(n)}(t) \xrightarrow{w} 0$ *in* $\mathcal{D}$.

Theorem 2.2.3 ([58], Theorem 3.3). *If* (2.1)–(2.4) *hold, then* $\hat{W}^{(n)}(t) \xrightarrow{w} \hat{Z}$ *in* $\mathcal{D}$.

Theorem 2.2.4 ([58], Theorem 3.4). *If* (2.1)–(2.4) *hold, then* $\hat{Q}^{(n)}(t) \xrightarrow{w} \lambda\hat{Z}$ *in* $\mathcal{D}$.

2.2.3 A Multi-Server Queue

Let $I > 1$ and $J > 1$. Consider a J-server queue with I superimposed renewal arrival processes. Assume the customers are served in the order of arrival by the first idle server. Let $Q_j(t)$ be the number of customers at time t that will be served at server j. $\tilde{A}_j(t)$ is the total number of customers that arrive in time $(0,t]$ and are served at server j. The actual service times at server j form

the sequence $\{v'_{j,m} : m \geq 1\}$, which is a subsequence of the original service times [1], so the elements $v'_{j,m}$ are still i.i.d.. The actual workload at server j is

$$Z_j(t) = \sum_{m=\tilde{A}_j(t)-[Q_j(t)-1]^+ +1}^{\tilde{A}_j(t)} v'_{j,m} + r_j(t),$$

where $r_j(t)$ is the residual service time of the customer being served by the j-th server at time t and $[Q_j(t) - 1]^+$ is the number of customers waiting for service at server j. The k-th arrival at time T_k has a sojourn time of

$$W(k) = \min_{1 \leq j \leq J} \{Z_j(T_k)\}.$$

Let $Q(t) = \sum_{j=1}^{J} Q_j(t)$ denote the queue length at time t. The total actual workload is

$$Z(t) = \sum_{m=A(t)-[Q(t)-J]^+ +1}^{A(t)} v'_m + \sum_{j=1}^{J} r_j(t),$$

where $\{v'_m\}$ are the actual service times of m arrivals.

Define the corresponding diffusion scaled processes in the same fashion as in (2.8)–(2.10).

Theorem 2.2.5 ([58], Theorem 5). *If* (2.1)–(2.4) *hold, then*

(i) If $c > -\infty$, *then* $\hat{Q}^{(n)}(t) \xrightarrow{w} \hat{Q} = RBM\left(c, \sum_{i=1}^{I} \lambda_i^3 \alpha_i^2 + \sum_{j=1}^{J} \mu_j^3 \beta_j^2\right)$ *in* $\mathcal{D}$, *as* $n \to \infty$. *If, in addition,* $J = 1$ *or all the servers are identical, then* $\hat{Z}^{(n)}(t) \xrightarrow{w} \hat{Q}/(J\mu_1)$ *and* $\hat{W}^{(n)}(t) \xrightarrow{w} \hat{Q}/(J\mu_1)$ *in* $\mathcal{D}$.

(ii) If $c = -\infty$, *then* $\hat{Q}^{(n)}(t) \xrightarrow{w} 0$ *in* $\mathcal{D}$, *as* $n \to \infty$. *If, in addition,* $J = 1$ *or all the servers are identical, then* $\hat{Z}^{(n)}(t) \xrightarrow{w} 0$ *and* $\hat{W}^{(n)}(t) \xrightarrow{w} 0$ *in* $\mathcal{D}$.

2.3 Complete Resource Pooling Condition

In this section, we introduce the concept of complete resource pooling (CRP). Intuitively, a system of m parallel queues which satisfies the CRP condition has

[1] Given the original service times $\{v_{j,\ell} : \ell \geq 1\}$ at server j, if there is no arrival waiting for service at the ℓ-th time, that time is ignored.

the same asymptotic behaviour as a single queue with m servers. Correspondingly, a multi-dimensional queue-length process collapses to one dimension in heavy traffic. This state space collapse enables the queue-length processes in different queues to be expressed in terms of the limit of the one-dimensional total queue-length process. This makes the system much easier to analyze, because for higher than one dimension, RBMs are not well understood.

For output-queued systems, Stolyar [63] has proposed the MinDrift(Q) routing rule which routes arrivals to a queue which has the minimal expected increment (or so-called drift) of the total holding cost (caused by the system workload). It has been proved that under CRP, MinDrift(Q) asymptotically minimizes the total workload among all routing rules, as long as the local scheduling rule is non preemptive (within the same customer type) and work-conserving, i.e., servers are not allowed to idle whenever there are customers waiting.

Here we follow Stolyar [63] to give a formal definition of the CRP condition and the limiting workload process of an output-queued system which is equipped with the MinDrift(Q) routing rule.

Consider a system with I types of customers and J servers. For each type i, the inter-arrival times form an i.i.d. sequence which has finite mean λ_i^{-1} and variance α_i^2. At each server j, the service times for type i customers form an i.i.d. sequence and have finite mean $\mu_{i,j}^{-1}$ and variance $\beta_{i,j}^2$. All arrival and service processes are assumed mutually independent.

Define a matrix $\Psi = (\psi_{i,j})_{I\times J}$, with all $\psi_{i,j} \geq 0$. Each element $\psi_{i,j}$ is the average rate at which server j's time is allocated to serve type i customers, in the long run. So the total utilization of server j is $\rho_j = \sum_{i=1}^{I} \psi_{i,j}$. The service capacity for type i customers is $\kappa_i = \sum_{j=1}^{J} \mu_{i,j}\psi_{i,j}$. Given the matrix Ψ, if in an output-queued system, customers of type i are routed to queue j at the average rate $(\mu_{i,j}\psi_{i,j})$, then the total service capacity for type i customers equals the mean arrival rate λ_i.

Define vectors $\lambda = [\,\lambda_1, \ldots, \lambda_I\,]$, $\kappa = [\,\kappa_1, \ldots, \kappa_I\,]$ and $\rho = [\,\rho_1, \ldots, \rho_J\,]$. The server utilization region is denoted by

$$\mathcal{U} = \{\rho \in \mathbb{R}_+^J : \kappa \geq \lambda\},$$

where $\mathbb{R}^J_+ = \{x \in \mathbb{R}^J : x \geq 0\}$ and the vector comparison is component-wise. The region $\mathcal{U}$ contains all the possible values of the server utilizations which satisfy the condition that the corresponding service capacity allocated to each task type is not less than the mean arrival rate of that type, a necessary condition for stabilizing the system. Let the vector $\xi^* = [\xi_1^*, \ldots, \xi_J^*]^T$ be the outer normal vector to the convex polyhedron $\mathcal{U}$ at point $\mathbf{1} \in \mathbb{R}^J$. (If $\mathbf{1}$ is outside the polyhedron $\mathcal{U}$, the corresponding system simply does not satisfy the CRP condition.) The inner product of vectors ρ and ξ^* is written as $\rho \cdot \xi^*$.

Theorem 2.3.1 ([63], Lemma 3, complete resource pooling). *The CRP condition for a fixed vector λ holds if and only if the following two conditions hold.*

(i) Vector $\mathbf{1} \in \mathbb{R}^J$ solves the problem

$$\begin{aligned} \min_{\rho \in \mathcal{U}} \quad & \rho \cdot \xi^* \\ s.t. \quad & \kappa \geq \lambda. \end{aligned}$$

(ii) The matrix Ψ which solves the linear system

$$\lambda = \kappa, \quad \rho = \mathbf{1} \tag{2.11}$$

is unique.

Let the matrix Ψ^* be a solution to (2.11). A graph $\mathcal{G}$ is constructed with nodes being customer types i and servers j, arcs (i, j) corresponding to a positive element $\psi^*_{i,j} > 0$. The CRP condition is equivalent to the condition that the graph $\mathcal{G}$ is a tree [35].

Define the capacity region

$$\mathcal{K} = \{\kappa \in \mathbb{R}^I_{++} : \rho \leq \mathbf{1}\},$$

where $\mathbb{R}^I_{++} = \{x \in \mathbb{R}^I : x > 0\}$. Given $\kappa = \lambda$, the region $\mathcal{K}$ contains all the mean arrival rate vectors for which the system can be stabilized. Let the vector $\nu^* = [\nu_1^*, \ldots, \nu_I^*]^T$ be the outer normal vector to the convex polyhedron $\mathcal{K}$ at the point λ. The CRP condition also implies $\mathbf{1} \cdot \xi^* = \lambda \cdot \nu^*$.

Let $Q_{i,j}(t)$ denote the number of type i customers at server j at time t, including the one in service. The workload estimated by the queue length at server j is defined as the Q-estimated workload

$$Z_j(t) = \sum_{i=1}^{I} \mu_{i,j}^{-1} Q_{i,j}(t).$$

The total server workload of the system is defined as

$$Z(t) = \sum_{j=1}^{J} \xi_j^* Z_j(t), \tag{2.12}$$

where the component ξ_j^* quantifies the workload contribution of server j. The total customer workload is defined as

$$Y(t) = \sum_{i=1}^{I} \nu_i^* \sum_{j=1}^{J} Q_{i,j}(t), \tag{2.13}$$

where ν_i^* quantifies the workload contribution of customer type i. For example, in a homogeneous server system with multiple task types, we have $\xi_j^* = 1/J$ for each server j and $\nu_i^* = 1/(J\mu_i)$, where μ_i is the mean service rate of type i tasks at each server. In general, we have $Z(t) \geq Y(t)$ for systems with multiple task types. They are equal under the CRP condition.

Finally, we state the heavy traffic optimality properties of the Min-Drift(Q) routing rule.

Assume that for each server j, a holding cost function $C_j(\zeta)$ $(\zeta \geq 0)$ is given and has the following properties [63]:

Assumption 2.3.1 (cost function properties).

1. *$C_j(\cdot)$ is continuous, strictly increasing and convex, with $C_j(0) = 0$;*

2. *The first derivative $C_j'(\cdot)$ is continuous and strictly increasing, with $C_j'(0) = 0$;*

3. *The second derivative $C_j''(\cdot)$ is strictly positive continuous in the open interval $(0, +\infty)$, with $C_j''(0) = \lim_{\zeta \downarrow 0} C_j''(\zeta)$ and $0 \leq C_j''(0) \leq +\infty$.*

The MinDrift(Q) rule routes a type i customer at arrival time t to a server j which satisfies

$$j \in \arg\min_{1 \le j \le J} \frac{C_j'(Z_j(t))}{\mu_{i,j}}.$$

Ties are broken arbitrarily.

Define the vector process

$$\mathbf{Z} = [\, Z_1(t), \ldots, Z_J(t) \,]. \tag{2.14}$$

The vector $\mathbf{Z}^* \in \mathbb{R}_+^J$ is called a fixed point if

$$[\, C_1'(Z_1^*(t)), \ldots, C_J'(Z_J^*(t)) \,] = c\,\xi^*$$

for some constant $c \ge 0$. For any $c > 0$, all components of $\mathbf{Z}^*$ are strictly positive. The Karush-Kuhn-Tucker (KKT) conditions [14] imply that a fixed point $\mathbf{Z}^*$ is the unique solution to the problem

$$\begin{aligned} \min_{\mathbf{Z}} \quad & f_0(\mathbf{Z}) = \sum_{j=1}^{J} C_j(Z_j(t)) \\ \text{s.t.} \quad & \mathbf{Z} \cdot \xi^* = \mathbf{Z}^* \cdot \xi^*. \end{aligned}$$

The fixed point $\mathbf{Z}^*$ minimizes the total cost among all vectors with the same server workload. The diffusion limit of $\mathbf{Z}^*$ is established in Theorem 2.3.2 as follows.

Suppose there is an output-queued system equipped with the Min-Drift(Q) routing rule and an arbitrary non preemptive, work-conserving local scheduling rule. Associated are the mean arrival rate vector λ which satisfies the CRP condition, the matrix Ψ^*, and the vectors ξ^* and ν^*. All of the queues are empty at the initial time $t = 0$. Consider a sequence of systems, indexed by n. For the n-th system, the inter-arrival times of task type i have mean $\left(\lambda_i^{-1}\right)^{(n)}$ and variance $\left(\alpha_i^2\right)^{(n)}$; the service times at server j for type i arrivals have mean $\mu_{i,j}^{-1}$ and variance $\beta_{i,j}^2$. We assume that the following conditions hold

$$\lim_{n\to\infty} \left(\alpha_i^2\right)^{(n)} = \alpha_i^2 \tag{2.15}$$

and

$$\sup_{n\geq 1, 1\leq i\leq I} \mathbf{E}\left[\left(u_{i,1}^{2+\epsilon}\right)^{(n)}\right] < \infty, \tag{2.16}$$

$$\mathbf{E}\left[v_{i,j,1}^{2+\epsilon}\right] \equiv c_{i,j}, \tag{2.17}$$

for some $\epsilon > 0$ and finite constant $c_{i,j}$. In addition the heavy traffic condition

$$\lim_{n\to\infty} \sqrt{n}\left(\lambda_i^{(n)} - \lambda_i\right) = b_i \tag{2.18}$$

for some finite constant b_i is assumed to be true for all $1 \leq i \leq I$. Since the vector $\lambda = (\lambda_i)_{1\times I}$ satisfies the CRP condition, (2.18) is equivalent to (2.4) in the sense that $b_i > -\infty$ (for all i) corresponds to $\rho = \mathbf{1}$.

Let $\hat{\mathbf{Z}}^{(n)}$, $\hat{Z}^{(n)}(t)$ and $\hat{Y}^{(n)}(t)$ denote the scaled versions of (2.14), (2.12) and (2.13), respectively, where the scaling is the same as in (2.8).

Theorem 2.3.2 ([63], Theorem 2(*i*)). *If* (2.15)–(2.18) *hold, then as* $n \to \infty$, $\hat{Z}^{(n)}(t) \xrightarrow{w} \hat{Z} = RBM(\theta, \sigma^2)$, *where*

$$\theta = \sum_{i=1}^{I} \nu_i^* b_i, \quad \sigma^2 = \sum_{i=1}^{I} (\nu_i^*)^2 \left[\lambda_i^3 \alpha_i^2 + \sum_{j=1}^{J} \psi_{i,j}^* \mu_{i,j}^3 \beta_{i,j}^2\right].$$

$\hat{\mathbf{Z}}^{(n)} \xrightarrow{w} \hat{\mathbf{Z}}$, *where for each* $t \geq 0$, *the vector* $\hat{\mathbf{Z}}$ *is a fixed point that is uniquely determined by* $\hat{\mathbf{Z}}(t) \cdot \xi^* = \hat{Z}(t)$.

Theorem 2.3.3 ([63], Theorem 3).

(i) If (2.15)–(2.18) *hold, then as* $n \to \infty$, $\hat{Y}^{(n)}(t) \xrightarrow{w} \hat{Z}$.

(ii) The MinDrift(Q) routing rule, in conjunction with an arbitrary work-conserving local scheduling rule, asymptotically minimizes the customer workload (2.13), *in an output-queued system. Specifically, the customer workload process* $\hat{Y}_\pi^{(n)}(t)$ *under a routing rule* π *other than MinDrift(Q) can be constructed on a common probability space with the RBM* $\hat{Z}$, *so that with probability 1, we have*

$$\liminf_{n\to\infty} \inf_{0\leq s\leq t} [\hat{Y}_\pi^{(n)}(s) - \hat{Z}(s)] \geq 0$$

for any time $t \geq 1$.

Chapter 3

Resource Allocation with Limited State Information

In this chapter, we propose several resource allocation policies, mainly routing policies which use the means of both the task inter-arrival times and the task processing times, for heterogeneous server systems with multiple task types. The task inter-arrival times follow *exponential* distributions, while the task processing times follow general continuous or discrete distributions. The routing policies are *dynamic* in the sense that they make routing decisions based on the current state information of the system, e.g., the queue lengths or the expected sojourn times. The proposed policies are designed to both maximize the capacity of the system and minimize the delay in the system by using a *subset* of global state information.

We study the routing policies in the following four sections. Section 3.1 introduces the system model. Section 3.2 gives Stolyar's MinDrift(Q) policy, which requires global state information to make routing decisions and thus is used as a reference for comparison. Section 3.3 presents our proposed policies, MinDrift Affinity ROuting policy (MARO) and its variants, that have several advantages over existing policies. In the last section, Section 3.4, special cases of applying MARO related policies to homogeneous server systems are studied. We study their applications in heterogeneous server systems in Chapter 5.

3.1 The Model

Define sets $\mathcal{I} = \{1, ..., I\}$ and $\mathcal{J} = \{1, ..., J\}$, where $I \geq 1$ and $J \geq 2$. Consider a system with I types of tasks and J servers. Type i tasks arrive according to a Poisson process with rate $\lambda_i > 0$. The processing times of type i tasks at server j are i.i.d. and form the sequence $\{v_{i,j,m} : m \geq 1\}$. Therefore, all inter-arrival time and processing time sequences are renewal processes. Moreover, they are assumed mutually independent. Both $\mu_{i,j} = 1/E[v_{i,j,1}]$ and $\beta_{i,j}^2 = Var[v_{i,j,1}]$ are assumed finite. We allow $\mu_{i,j} = 0$, which implies server j is physically incapable of processing type i tasks. In addition, each task type can be processed by at least one server, i.e., $(\forall i \in \mathcal{I})(\exists j \in \mathcal{J})\ \mu_{i,j} > 0$. Collectively, we define the first-order primitives: vector $\lambda = (\lambda_i)_{1\times I}$ and matrix $\mu = (\mu_{i,j})_{I\times J}$, and second-order primitive: matrix $\beta = (\beta_{i,j}^2)_{I\times J}$.

The topology of the system is that servers work in parallel at each task type and each server maintains its own queue with a buffer of infinite size. A task must be dispatched (by a dispatcher) to one of the servers j immediately upon arrival. After being assigned, the task stays in queue j until it is processed and leaves the system upon completion of processing. Such a model is an "output-queued" (OQ) system.

The processing discipline satisfies two conditions:

Condition 3.1.1. *The local scheduling rule at each server is non-preemptive within each task type. Preemption of service or server sharing is allowed only for tasks of different types. A server is not allowed to idle while there are tasks waiting in its queue.*

Condition 3.1.2. *A task's processing time is not known until its completion, though the task type and thus its processing time distribution is known to the dispatcher and the servers.*

Let $Q_{i,j}(t)$ denote the number of type i tasks at queue j (including the one in process) at time t. The estimate of the unfinished processing time at server j at time t is given by the Q-estimated workload

$$Z_j(t) = \sum_{i\in\mathcal{I}} \mu_{i,j}^{-1} Q_{i,j}(t). \tag{3.1}$$

We use Z_j to denote the estimate calculated at each arrival time (equivalently, when a routing decision is to be made). In addition, each server j is given a holding cost function $C_j(\cdot)$ which satisfies Assumption 2.3.1. An example cost function is a single-term polynomial of the form $C_j(z) = c_j z^n$, where $c_j > 0$ and $n > 1$. We allow different cost functions at different servers.

3.2 MinDrift(Q) Policy

Stolyar [63] introduced the MinDrift(Q) rule, which attempts to minimize the average increase rate of the aggregate cost $\sum_{j\in\mathcal{J}} C_j(Z_j)$, due to placement of new tasks to the servers.

Policy 3.2.1 (MinDrift(Q)). *A type i arrival is dispatched to a queue j satisfying*

$$j \in \arg\min_{j\in\mathcal{J}} \frac{C_j'(Z_j)}{\mu_{i,j}}.$$

Ties are broken arbitrarily, for example, in favour of the smallest queue index.

The value $C_j'(Z_j)/\mu_{i,j}$ approximates the expected increment of the aggregate cost (caused by routing one type i task to server j). The MinDrift(Q) policy has been proved, in the heavy traffic regime, to asymptotically minimize the holding cost rate (or the so-called "drift") at all times, when the first-order primitives λ and μ satisfy the complete resource pooling (CRP) condition (see Theorem 2.3.1).

The fact that this policy satisfies such an optimality property means that it also stabilizes the system [62, 63]. This makes for a desirable starting point, but we need to be careful about naively implementing it. Policy 3.2.1 suffers from the fact that to make a routing decision, the dispatcher must know, in addition to the μ matrix, the current state information of *all* queues. Also, the optimality property of Policy 3.2.1 is obtained under a heavy traffic assumption where the system load approaches 100 percent. When one backs off from the heavy traffic condition, we will see (e.g., in Section 5.2.3) that there is room for making bad routing decisions, which in turn can significantly degrade performance.

Motivated by reducing the overhead of obtaining state information, as well as maintaining high performance levels under a range of traffic conditions, we propose in the next subsection a set of routing policies, which adapt Policy 3.2.1 to address these concerns.

3.3 MinDrift Affinity Routing Policies

3.3.1 MARO

This Mindrift Affinity ROuting (MARO) policy, along with its variants proposed in the following subsections, involves solving a linear programming (LP) problem whose goal is to minimize server utilizations (equivalently, maximizing the system capacity). Since in general, server utilizations (associated with first-order primitives) have a significant impact on performance metrics such as mean queue lengths and sojourn times in the system, it seems reasonable that policies which make the routing decisions by using the solutions to the LP should be expected to achieve comparatively good performance.

The Static Allocation LP

Let the scalar ρ denote the long-run utilization of the busiest server. Define the matrix $\Psi = (\psi_{i,j})_{I \times J}$ where the elements $\psi_{i,j}$ are interpreted as the average rate at which the j-th server's time is allocated to process type i tasks. The static (processing time) allocation problem can be described by the following LP:

$$\min_{(\rho,\Psi)} \quad \rho \tag{3.2}$$

$$\text{s.t.} \quad \sum_{j=1}^{J} \psi_{i,j}\mu_{i,j} = \lambda_i, \quad \forall i \in \mathcal{I}, \tag{3.3}$$

$$\sum_{i=1}^{I} \psi_{i,j} \leq \rho, \quad \forall j \in \mathcal{J}, \tag{3.4}$$

$$\rho \geq 0, \quad \psi_{i,j} \geq 0, \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}.$$

Several observations can be made about this LP:

Firstly, LP (3.2) always has a solution, since no upper bound constraint is put on ρ. However, the physical meaning of ρ requires that its value not exceed one. Therefore, if the first-order primitives (λ, μ) are such that LP (3.2) has an optimal solution $\rho^* > 1$, it means that each mean arrival rate λ_i should be scaled down by a factor of $1/\rho^*$ at least, otherwise the system cannot be stabilized. If $\rho^* \leq 1$, the reciprocal of the optimal value ρ^* is the maximum factor by which each arrival rate λ_i can be increased so that the system can still be stabilized [59]. In that case, LP (3.2) also maximizes the system capacity.

Secondly, constraint (3.3) enforces the processing capacity for each task type is not less than the task's arrival rate, which provides a necessary condition for stabilizing the system. On the other hand, (3.3) requires the processing capacity to not exceed a task's arrival rate. If a task type i was over-allocated with service times, we could reduce the values of the $\psi_{i,j}$'s until constraint (3.3) is satisfied, while one or more of the constraints in (3.4) is not tight at optimum. If a constraint in (3.4) is tight at optimum, the corresponding server is considered as a bottleneck. In this thesis, we focus on a *bottleneck system* which meets the following assumption:

Assumption 3.3.1. *The tuple of primitives* (λ, μ) *is such that LP* (3.2) *has solutions* (ρ^*, Ψ^*) *satisfying*

$$\sum_{i=1}^{I} \psi_{i,j}^* = \rho^* \leq 1, \quad \forall j \in \mathcal{J}. \tag{3.5}$$

This means that if $\sum_{i=1}^{I} \psi_{i,j}^* < \rho^*$ for any server j, that server is underutilized and will be removed from our model (together with the portion of tasks that were routed to this server).

One special case of first-order primitives which satisfy the above assumption is that all mean processing rates $\mu_{i,j}$ are positive. This requires that all servers are capable of processing all task types. Using Proposition 3.3.1, we will show that the solutions of LP (3.2) remain the same even if a number of fully-flexible servers are substituted by servers which are only capable of processing a subset of different task types. This leads to our third observation.

Let N_p be the number of positive elements of a solution Ψ^*. We have

Proposition 3.3.1. *There exists a solution Ψ^*, whose associated N_p satisfies* $\max(I, J) \leq N_p \leq I + J - 1$.

Proof. The first inequality follows the assumptions made on the model in Section 3.1, otherwise there exists either at least one task type that is not processed by any servers or at least one idle server.

The second inequality results from the generation of the basic optimal solution [11] of LP (3.2). Since the LP always has a non-zero optimal solution which contains $IJ + 1$ variables, then a basic optimal solution exists. The solution contains $I+J$ linearly independent (basic) variables which correspond to $I + J$ constraints and $IJ+1-(I+J)$ (nonbasic) variables which are zeros. Since one of the basic variables is always $\rho^* > 0$, the matrix Ψ^* has at least $IJ + 1 - (I + J)$ elements equal to zero. Equivalently, the number of positive elements is at most $I + J - 1$. □

Given system A with a full matrix ${}^A\mu$ where ${}^A\mu_{i,j} > 0, \forall i \in \mathcal{I}, \forall j \in \mathcal{J}$ and the corresponding solution ${}^A\Psi^*$, we may construct system B with matrix ${}^B\mu$ where

$$ {}^B\mu_{i,j} = \begin{cases} {}^A\mu_{i,j}, & \text{if } {}^A\psi^*_{i,j} > 0; \\ 0, & \text{if } {}^A\psi^*_{i,j} = 0; \end{cases} $$

and obtain the corresponding solution ${}^B\Psi^*$. It is easy to verify ${}^B\Psi^* = {}^A\Psi^*$. So the two systems will have the same performance when they use the following affinity routing policy, while system B only requires each server have limited flexibility in processing different task types. This is desirable when it is costly to maintain highly flexible servers.

The Affinity Routing Policy

Given a solution Ψ^*, we define for each task type i the set S_i of servers that will potentially process that type, i.e.,

$$ S_i = \{j : \psi^*_{i,j} \neq 0, j \in \mathcal{J}\}. \tag{3.6} $$

It is true that $|S_i| > 0$ for all $i \in \mathcal{I}$, since any type i tasks will get processed by at least one server. Thus, we propose the MARO policy as follows:

Policy 3.3.1 (MARO). *A type $\imath$ arrival is dispatched to a queue j satisfying*

$$j \in \arg\min_{j \in S_\imath} \frac{C_j'(Z_j)}{\mu_{\imath,j}}.$$

Ties are broken randomly with equal probabilities.

The MARO policy has the following properties:

Firstly, it not only aims to maximize the capacity in the long term by using the solution to the LP (3.2), but also in the short term shifts the workload between servers to avoid congestion by using the current state information at each arrival time.

Secondly, MARO differs from MinDrift(Q) in that the dispatcher does not have to know the current workload information of *all* servers for each task type (at each arrival time). It does require that the arrival rates of all task types must be estimated in order to determine the set $S_\imath$. Nevertheless, MARO is robust to perturbations in the mean arrival rates in the sense that increasing each $\lambda_\imath$ by the same factor ε_λ does not change the sets $S_\imath$. This can be seen by multiplying ε_λ on both sides of (3.3). On the other hand, the arrival rate information is also needed for MinDrift(Q) when checking if the system can be stabilized.

Thirdly, MARO benefits as the size of the set $S_\imath$ is smaller than the total number of servers J for most task types i, i.e., Ψ^* is a sparse matrix. From Proposition 3.3.1, it is can been seen that when J increases, Ψ^* becomes sparser. To measure this, we introduce the idea of "discount amount of state information" as follows.

Let N_s be the average number of servers from which MARO acquires state information for each arrival. We have

$$N_s = \sum_{\imath:|S_\imath|>1} \frac{\lambda_\imath}{\tilde{\lambda}} |S_\imath|, \tag{3.7}$$

where $\tilde{\lambda} = \sum_{i \in \mathcal{I}} \lambda_i$. If $|S_i| = 1$, type i tasks join a single server without acquiring state information. Definition (3.6) implies that N_s is a function of J. For MinDrift(Q), the corresponding number of servers is J (assuming that all $\mu_{i,j}$ are positive). Thus, using MinDrift(Q) as a reference, we define the

discount of the average required state information for a routing decision by

$$Discnt_{(MARO)} = \left(1 - \frac{N_s(J)}{J}\right) \times 100\%. \tag{3.8}$$

Given Proposition 3.3.1 and $I > 1$, we have

Proposition 3.3.2. *If $N_p = \sum_{i\in\mathcal{I}} |S_i| = I + J - 1$ and $\lambda_i/\tilde{\lambda} = 1/I$ for all $i \in \mathcal{I}$, then $Discnt_{(MARO)}$ is monotonically increasing in J.*

Proof.

Let $J' = \alpha J$, $\alpha > 1$ and $\sum_{i\in\mathcal{I}} |S_i'| = I + J' - 1$. Let $f(J) = N_s(J)/J$, we have

$$\begin{aligned} f(J') - f(J) &= \frac{1}{J}\sum_{i\in\mathcal{I}} \frac{\lambda_i}{\tilde{\lambda}} \left(\frac{|S_i'|}{\alpha} - |S_i|\right) \\ &= \frac{1}{IJ}\sum_{i\in\mathcal{I}} \left(\frac{|S_i'|}{\alpha} - |S_i|\right) \\ &= \frac{I-1}{IJ}\left(\frac{1}{\alpha} - 1\right) < 0. \end{aligned}$$

So $f(J)$ is monotonically decreasing in J, which implies $Discnt_{(MARO)}$ is monotonically increasing in J.

□

Proposition 3.3.2 implies that MARO is more advantageous in systems with a large number of servers, when the arrival rates of different task types are equal. It is noted, however, that equal arrival rates are not necessary for $Discnt_{(MARO)}$ to be monotonically increasing in J. We will discuss the effect of a single dominant task type in Section 3.3.5.

Fourthly, although MARO is more favourable when the value N_p is small (so that N_s is small), to have MARO take advantage of the state information in order to shift workload between servers, we do not want N_p too small. We construct a graph $\mathcal{G}_\Psi$ which has nodes being arrival types i and servers j and arcs (i, j) being the routing activities. The graph $\mathcal{G}_\Psi$ is associated with the matrix Ψ^* in that an arc (i, j) corresponds to a non-zero element $\psi^*_{i,j}$. Therefore, if $N_p < I + J - 1$ (which corresponds to a degenerate basic

solution to the LP), then $\mathcal{G}_\Psi$ is not a connected graph, which means there is at least one server whose load cannot be shifted to the others. The extreme case is that when N_p reaches the lower bound $\max(I, J)$, no workload can be shifted among the servers. This will reduce MARO to a static routing policy (which does not need state information for any task type). If $N_p > I + J - 1$, then $\mathcal{G}_\Psi$ contains at least one ring. This implies that LP (3.2) has infinitely many solutions, since we can perturb a solution along the arcs of a ring. On the other hand, we may use such a perturbation to obtain a solution Ψ^* such that $N_p = I + J - 1$, i.e., graph $\mathcal{G}_\Psi$ represents a tree. This not only reduces the amount of required state information, but also enables workload to be shifted among all servers. We now provide two examples to illustrate this concept.

Example 3.3.1. *Let*

$$\lambda = \begin{bmatrix} a & b \end{bmatrix}, \quad \mu = \begin{bmatrix} a & a \\ b & b \end{bmatrix}.$$

One solution of LP (3.2) *is*

$$\rho^* = 1, \quad \Psi^* = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix},$$

with $N_p > I + J - 1 = 3$. *By perturbing* Ψ^* *along the arcs of the ring* $(1,1) - (2,1) - (2,2) - (1,2)$, *we have*

$$\Psi^*_\varepsilon = \begin{bmatrix} 0.5 + \varepsilon_\mu & 0.5 - \varepsilon_\mu \\ 0.5 - \varepsilon_\mu & 0.5 + \varepsilon_\mu \end{bmatrix}.$$

When $\varepsilon_\mu = 0.5$, *we have* Ψ^* *equal to the identity matrix and* $N_p = 2$. *This reduction of* N_p *is not desirable for MARO, since workload cannot be shifted between the two servers.*

Example 3.3.2. *Let*

$$\lambda = \begin{bmatrix} \frac{a}{2} & \frac{b}{3} \end{bmatrix}, \quad \mu = \begin{bmatrix} a & a \\ b & b \end{bmatrix}.$$

One solution of LP (3.2) *is*

$$\rho^* = \frac{5}{12}, \quad \Psi^*_1 = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{6} \end{bmatrix}.$$

Instead of directly applying Ψ_1^* *to the MARO policy, we perturb* Ψ_1^* *and obtain*

$$\Psi_2^* = \begin{bmatrix} \frac{5}{12} & \frac{1}{12} \\ 0 & \frac{1}{3} \end{bmatrix},$$

with $N_p = I + J - 1$. *Using* Ψ_2^*, *MARO requires the minimum amount of state information while still allowing workload to be shifted between the two servers.*

Finally, we use Theorem 3.3.1 to show that under the same assumptions on the primitives (λ, μ), MinDrift(Q) and MARO have the same optimality properties in heavy traffic, so from that standpoint we have not lost anything by using less state information. We will perform simulation studies in Section 5.2.3, which suggest that under realistic load MARO can outperform MinDrift(Q) in heterogeneous server systems.

Assume that the following conditions hold for the primitives (λ, μ):

Assumption 3.3.2. *LP* (3.2) *has a unique solution* (ρ^*, Ψ^*), *where* $\rho^* = 1$ *and* $\sum_{i=1}^{I} \psi_{i,j}^* = 1$, *for all* $j \in \mathcal{J}$.

Assumption 3.3.3. $\sum_{i \in \mathcal{I}} |S_i| = I + J - 1$, *where the set* S_i *is defined in* (3.6).

Assumption 3.3.2, which is stronger than Assumption 3.3.1, implies that the system is under heavy traffic. Assumption 3.3.3 implies that the graph $\mathcal{G}_{\Psi^*}$ associated with the matrix Ψ^* represents a tree, so the workload of each server can be shifted to the others. Then using Theorem 2.3.1 (complete resource pooling), we have that the CRP condition holds for the mean arrival rate vector λ.

We rewrite (2.13) to define the total customer workload

$$Y(t) = \sum_{i \in \mathcal{I}} \nu_i^* \sum_{j \in \mathcal{J}} Q_{i,j}(t), \tag{3.9}$$

where ν_i^* quantifies the contribution of type i arrivals to the total workload

and is the solution to the dual of LP (3.2),

$$\begin{aligned} \max_{(\nu,\xi)} \quad & \sum_{i=1}^{I} \nu_i \lambda_i \\ \text{s.t.} \quad & \nu_i \mu_{i,j} \leq \xi_j, \quad \forall j \in \mathcal{J}, \forall i \in \mathcal{I}, \\ & \sum_{j=1}^{J} \xi_j \leq 1, \\ & \nu_i \geq 0, \forall i \in \mathcal{I} \text{ and } \xi_j \geq 0, \forall j \in \mathcal{J}. \end{aligned} \tag{3.10}$$

Vector $\nu = (\nu_i^*)_{1\times I}$ corresponds to constraint (3.3), so we have $\nu_i^* > 0$ for all $i \in \mathcal{I}$. In addition, we have $\nu_i^* \mu_{i,j} = \xi_j^*$ if $\psi_{i,j}^* \neq 0$, $\nu_i^* \mu_{i,j} < \xi_j^*$ if $\psi_{i,j}^* = 0$ and $\sum_{j=1}^{J} \xi_j^* = 1$, $\xi_j^* > 0$ for all $j \in \mathcal{J}$, under Assumptions 3.3.2 and 3.3.3.

To define the heavy traffic regime, we consider a sequence of systems indexed by n. For the n-th system, the inter-arrival times of task type i are exponentially distributed with mean $\left(\lambda_i^{-1}\right)^{(n)}$; the processing times at server j for type i arrivals have mean $\mu_{i,j}^{-1}$ and variance $\beta_{i,j}^2$. We assume that the following conditions hold

$$\lim_{n\to\infty} \lambda_i^{(n)} = \lambda_i, \tag{3.11}$$

where the vector $\lambda = (\lambda_i)_{1\times I}$ satisfies the CRP condition (defined in Section 2.3) and

$$\mathbf{E}\left[v_{i,j,1}^{2+\epsilon}\right] \equiv d_{i,j} < \infty, \tag{3.12}$$

for some $\epsilon > 0$ and constant $d_{i,j}$, where $\{v_{i,j,m} : m \geq 1\}$ is a sequence of i.i.d. random variables formed by the processing times at queue j for task type i. In addition, the heavy traffic condition

$$\lim_{n\to\infty} \sqrt{n}\left(\lambda_i^{(n)} - \lambda_i\right) = b_i \tag{3.13}$$

for some finite constant b_i is assumed to be true for all $i \in \mathcal{I}$.

Let $Q_{i,j}^{(n)}(t)$ be the queue length process of type i arrivals at server j at time t. We define its diffusion scaled version

$$\hat{Q}_{i,j}^{(n)}(t) = \frac{1}{\sqrt{n}} Q_{i,j}^{(n)}(nt). \tag{3.14}$$

Consequently, the diffusion scaled total workload process derived from (3.9) is denoted as $\hat{Y}^{(n)}(t)$. For the same sequence of systems on which MinDrift(Q) (Policies 3.2.1) and MARO (Policy 3.3.1) are operating, we denote the diffusion scaled total workload as $\hat{Y}_d^{(n)}(t)$ and $\hat{Y}_a^{(n)}(t)$, respectively.

To show that the two policies have the same optimality properties, we have the following results:

Theorem 3.3.1.

(i) For the MinDrift(Q) policy, the diffusion scaled total customer workload process, $\hat{Y}_d^{(n)}(t)$, converges weakly to a one-dimensional reflected Brownian motion $\hat{Y}_d$. To be precise, $\hat{Y}_d^{(n)}(t) \xrightarrow{w} \hat{Y}_d = RBM(\theta,\ \sigma^2)$, as $n \to \infty$, where

$$\begin{aligned} \theta &= \sum_{i\in\mathcal{I}} \nu_i^* b_i, \\ \sigma^2 &= \sum_{i\in\mathcal{I}} (\nu_i^*)^2 \left[\lambda_i + \sum_{j\in\mathcal{J}} \psi_{i,j}^* \mu_{i,j}^3 \beta_{i,j}^2\right]. \end{aligned} \tag{3.15}$$

(ii) For the MARO policy, $\hat{Y}_a^{(n)}(t) \xrightarrow{w} \hat{Y}_d$.

Proof.

(i) Under the conditions (3.11) – (3.13), a direct application of Theorem 2.3.3(i) yields the result. Furthermore, Theorem 2.3.3(ii) implies that the MinDrift(Q) routing policy minimizes the total workload (3.9) and is asymptotically optimal among all service disciplines which satisfy Conditions 3.1.1 and 3.1.2 described in Section 3.1.

(ii) Let the tuple $((\mu_{i,j})_d, (\beta_{i,j}^2)_d)$ characterize the processing time distribution of the system on which MinDrift(Q) is operating. For the system to which MARO is applied, we have

$$(\mu_{i,j})_a = \begin{cases} (\mu_{i,j})_d, & \text{if } \psi_{i,j}^* \neq 0, \\ 0, & \text{if } \psi_{i,j}^* = 0, \end{cases}$$

and

$$(\beta_{i,j}^2)_a = \begin{cases} (\beta_{i,j}^2)_d, & \text{if } \psi_{i,j}^* \neq 0, \\ 0, & \text{if } \psi_{i,j}^* = 0. \end{cases}$$

Consequently, the same mean arrival rate vector λ satisfies the CRP condition when MARO is applied. Then using the same reasoning as in Part (i), we have the result.

□

3.3.2 MARO$-2/k$

To reduce further the state information required in making server assignment decisions, one way is to choose for each arrival just two queues from the set S_i and then compare that pair of holding cost drifts.

Policy 3.3.2 (MARO$-2/k$). *A type i arrival is dispatched to one of the two queues (j_1, j_2) chosen from S_i such that the arrival joins a queue j satisfying*

$$j \in \arg \min_{j \in \{j_1, j_2\}} \frac{C_j'(Z_j)}{\mu_{i,j}}.$$

If $|S_i| > 2$, the two queues j_1 and j_2 are chosen with probabilities $p_{j_1} = r_{i,j_1}/\tilde{r}_i$ and $p_{j_2} = r_{i,j_2}/(\tilde{r}_i - r_{i,j_1})$, respectively, where $r_{i,j} = \psi_{i,j}^* \mu_{i,j}$ and $\tilde{r}_i = \sum_{j \in S_i} r_{i,j}$.

Policy 3.3.2 only needs the state information of two queues (j_1, j_2). Since (3.3) holds, we have $\tilde{r}_i = \lambda_i$ and $r_{i,j}$ denotes the (optimal) average rate at which type i tasks are routed to server j. Therefore, the first queue j_1 is picked with a probability to achieve this optimum and the second queue j_2 is chosen with a conditional probability given j_1 is picked. Since $\sum_{(j_1, j_2) \in S_i} p_{j_1} p_{j_2} = 1$, the probabilities given in Policy 3.3.2 are well defined.

Similarly, we can define the discount of the average required state information by

$$Discnt_{(2k)} = \left(1 - \frac{N_{s_{(2k)}}}{J}\right) \times 100\%, \tag{3.16}$$

where

$$N_{s_{(2k)}} = 2 \sum_{i:|S_i|>1} \frac{\lambda_i}{\tilde{\lambda}} \tag{3.17}$$

is the average number of servers from which MARO$-2/k$ acquires state information for each arrival. It is noted that the worst case of $Discnt_{(2k)}$ is

$(J-2)/J \times 100\%$. This implies that when the number of servers grows, the discount $Discnt_{(2k)}$ increases independently of the structure of Ψ^*, which is even more favourable than MARO.

A special case of the system to which Policy 3.3.2 is applied is the "supermarket" model with J identical servers (processing times exponentially distributed, mean one, FIFO service discipline) and a Poisson arrival process with rate $J\lambda_1$ (for only one task type, $\lambda_1 < 1$). In this case, $\psi^*_{1,j} = 1$ for all $j \in \mathcal{J}$. An arrival thus randomly (with equal probabilities) chooses two of the J servers and joins the queue of the server with the shorter queue length. Mitzenmacher [51] analyzed such a system and found that when λ_1 approaches 1 (which implies the system is under heavy traffic) and $J \to \infty$, there is an exponential improvement in the mean sojourn time (over choosing only one out of J servers randomly), while increasing the number of choices for an arrival results in only a constant improvement over two choices. This suggests that a similar degree of improvement might be expected for MARO and MARO$-2/k$ over a static routing policy, although the "power of two choices" has not been analyzed rigorously for heterogeneous systems.

3.3.3 MARO–flex

The MARO$-2/k$ policy in the long run still requires state information of all queues in the set S_i for all type i tasks. If either in the long run only a certain proportion of time is available for the dispatcher to acquire state information from the distributed server queues, or only a certain proportion of type i tasks can afford (or need) to be routed to a queue based on dynamic choice, we can introduce another way of limiting the required state information, as proposed through the following routing policy.

Policy 3.3.3 (MARO–flex). *Given* $0 < q_i \leq 1$, *for each task type* i, $100(1-q_i)$ *percent are assumed to be dedicated arrivals; the remaining* $100q_i$ *percent are assumed to be flexible arrivals. A dedicated arrival is routed to queue* j $(j \in S_i)$ *with probability*

$$p^d_{i,j} = \psi^*_{i,j}\mu_{i,j}/\tilde{r}_i, \tag{3.18}$$

where $\tilde{r}_i = \sum_{j_k \in S_i} \psi^*_{i,j_k} \mu_{i,j_k}$ *is the total processing capacity for type* i *tasks. A flexible arrival is routed to queue* j *using Policy 3.3.1, MARO.*

Since (3.3) holds, we have $\tilde{r}_i = \lambda_i$ and $r_{i,j} := \psi^*_{i,j}\mu_{i,j}$ denotes the (optimal) average rate at which type i tasks are routed to server j. In the proof of Theorem 3.3.2, we will see that both dedicated and flexible arrivals are routed to achieve this optimum.

MARO can be considered as a special case of MARO−flex with $q_i = 1$ for all task types i. Therefore, MARO−flex has the same properties as MARO has. If $q_i = 0$ for all task types i, MARO−flex becomes a static routing policy.

To quantify the reduction in the amount of state information used by the MARO−flex Policy, we define the discount

$$Discnt_{(flx)} = \left(1 - \frac{N_{s_{(flx)}}}{J}\right) \times 100\%, \tag{3.19}$$

where by (3.7),

$$N_{s_{(flx)}} = \sum_{i:|S_i|>1} \frac{q_i \lambda_i}{\tilde{\lambda}} |S_i|$$

is the average number of servers from which MARO−flex acquires state information. We have the same result as stated in Proposition 3.3.2, that $Discnt_{(flx)}$ is monotonically increasing in J in the case where the arrival rates of all task types are equal.

Heavy Traffic Analysis

We use Theorem 3.3.2 to show that in heavy traffic, a small amount of flexibility in the MARO−flex policy should give close to the performance improvement given by 100 percent flexibility (i.e., the MARO policy). When MARO−flex is applied to the same sequence of systems considered in Theorem 3.3.1, we denote the diffusion scaled total workload process by

$$\hat{Y}^{(n)}_f(t) = \sum_{i \in \mathcal{I}} \nu^*_i \sum_{j \in \mathcal{J}} \hat{Q}^{(n)}_{f,i,j}(t), \tag{3.20}$$

where $\hat{Q}^{(n)}_{f,i,j}(t)$ is the MARO−flex version of (3.14). Then we have have

Theorem 3.3.2. $\hat{Y}_f^{(n)}(t) \xrightarrow{w} \hat{Y}_d$, *where the diffusion limit is independent of both* q_i *and* $p_{i,j}^d$.

Since the difference between the two unscaled sequences of processes $Y_a(t)$ and $Y_f(t)$ is of the order $o(\sqrt{n})$, Theorems 3.3.1(*ii*) and 3.3.2 suggest that the performance of the systems should be relatively close (particularly for high loads).

Proof.

Suppose the elements of S_i are indexed as $\psi^*_{i,j_1}, ..., \psi^*_{i,j_{K_i}}$, where $j_1 < \cdots < j_{K_i}$ and $K_i = |S_i|$, then (3.6) is equivalent to

$$S_i = \{j_k : \psi^*_{i,j_k} \neq 0, k = 1, ..., K_i\} \tag{3.21}$$

and (3.18) is equivalent to

$$p^d_{i,j_k} = \psi^*_{i,j_k} \mu_{i,j_k} / \tilde{r}_i. \tag{3.22}$$

Define the index set $\hat{\mathcal{I}} = \bigcup_{i \in \mathcal{I}} \hat{S}_i^d \cup \bigcup_{i \in \mathcal{I}} \hat{S}_i^f$, where $\hat{S}_i^d = \{(i, j_k) : j_k \in S_i\}$ and $\hat{S}_i^f = \{(i, S_i)\}$. Define the set $\mathcal{L} = \{1, ..., |\hat{\mathcal{I}}|\}$ and a one-to-one function $f_1 : \hat{\mathcal{I}} \to \mathcal{L}$ of the form

$$\begin{cases} f_1(1, j_k) &= k, & k \in \{1, ..., K_1\}, \\ f_1(1, S_1) &= K_1 + 1, \end{cases}$$

and

$$\begin{cases} f_1(i, j_k) &= \sum_{n=1}^{i-1} K_n + k, & i > 1, k \in \{1, ..., K_i\}, \\ f_1(i, S_i) &= \sum_{n=1}^{i} K_n + 1, & i > 1. \end{cases}$$

The arrival rate vector $\hat{\lambda} = (\hat{\lambda}_\ell)_{1 \times |\hat{\mathcal{I}}|}$ has elements grouped with respect to each task type i and indexed by subscripts ℓ_i. In each group i, the arrival rates are

$$\hat{\lambda}_{\ell_i} = \begin{cases} (1 - q_i)\lambda_i p^d_{i,j_k}, & \text{for the dedicated arrivals,} \\ q_i \lambda_i, & \text{for the flexible arrivals,} \end{cases} \tag{3.23}$$

where the indices ℓ_i are determined by

$$\ell_i = f_1(x) := \begin{cases} i_k^d, & x \in \hat{S}_i^d, \\ i^f, & x \in \hat{S}_i^f. \end{cases} \tag{3.24}$$

So for each type i, we have

$$\begin{cases} \sum_{k=1}^{|S_i|} \hat{\lambda}_{i_k^d} & = (1-q_i)\lambda_i, \\ \hat{\lambda}_{if} & = q_i\lambda_i. \end{cases}$$

Define the matrix $\Phi = (\phi_{\ell,j})_{|\hat{\mathcal{I}}|\times J}$, where the quantity $(\phi_{\ell,j}\mu_{i,j})$ is the average rate at which arrivals of subtype ℓ (within task type i) are routed to queue j. It consists of I sub-matrices $\Phi_1, ..., \Phi_I$, each corresponding to a task type i ($i \in \mathcal{I}$). Each sub-matrix (with only non-zero elements shown) is

$$\Phi_i = \begin{bmatrix} \Phi_i^d \\ \hline \Phi_i^f \end{bmatrix} = \left[\begin{array}{cccc} \phi_{i_1^d,j_1} & & & \\ & \ddots & & \\ & & & \phi_{i_{K_i}^d,j_{K_i}} \\ \hline \phi_{if,j_1} & \phi_{if,j_2} & \cdots & \phi_{if,j_{K_i}} \end{array}\right]_{(K_i+1)\times J},$$

where the row indices i_k^d and i^f (denoted collectively as ℓ_i) are determined using (3.24). Let the non-zero elements ϕ_{ℓ_i,j_k} be

$$\begin{cases} \phi_{i_k^d,j_k} & = (1-q_i)\psi^*_{i,j_k}, \\ \phi_{if,j_k} & = q_i\psi^*_{i,j_k}, \end{cases} \tag{3.25}$$

for all $k \in \{1, ..., K_i\}$. Given $0 < q_i \leq 1$, all the elements $\phi_{\ell,j}$ are positive for *flexible* arrivals. This combined with Assumption 3.3.3 (the matrix Ψ^* representing a tree), implies that the matrix Φ also represents a tree, whose nodes are arrival subtypes ℓ and queues j and arcs (ℓ, j) are the actual routing activities.

From (3.22), (3.23) and (3.25), we have for each task type i a set of equalities

$$\begin{cases} \phi_{i_k^d,j_k}\mu_{i,j_k} & = \hat{\lambda}_{i_k^d}, & \forall k \in \{1, ..., K_i\}, \\ \sum_{k=1}^{K_i} \phi_{if,j_k}\mu_{i,j_k} & = \hat{\lambda}_{if}, & \\ \sum_{\ell_i} \phi_{\ell_i,j_k} & = \psi^*_{i,j_k}, & \forall k \in \{1, ..., K_i\}. \end{cases} \tag{3.26}$$

Assumption 3.3.2 and (3.26) imply that $\phi_{\ell,j}$ defined in (3.25) is a unique solution to the linear system

$$\begin{cases} \sum_{j\in\mathcal{J}} \phi_{\ell,j}\mu_{i,j} = \hat{\lambda}_\ell, & \forall \ell \in \mathcal{L}, \\ \sum_{\ell\in\mathcal{L}} \phi_{\ell,j} = 1, & \forall j \in \mathcal{J}. \end{cases} \tag{3.27}$$

Therefore, from Theorem 2.3.1, the CRP condition holds for the arrival rate vector $\hat{\lambda}$ defined in (3.23).

For each task type i, let

$$\hat{b}_{\ell_i} = \begin{cases} (1-q_i)b_i p^d_{i,j_k}, & \text{for the dedicated arrivals,} \\ q_i b_i, & \text{for the flexible arrivals,} \end{cases} \tag{3.28}$$

so that $\sum_{\ell_i} \hat{b}_{\ell_i} = b_i$. From (3.13) and (3.28), we have

$$\lim_{n\to\infty} \sqrt{n}\left(\hat{\lambda}^{(n)}_{\ell_i} - \hat{\lambda}_{\ell_i}\right) = \hat{b}_{\ell_i},$$

where $\hat{\lambda}_{\ell_i}$ is defined in (3.23).

Let $\hat{\nu}^*_{\ell_i}$ quantify the contribution of subtype ℓ_i arrivals to the total customer workload. We define the diffusion scaled total customer workload at time t

$$\hat{Y}^{(n)}_F(t) = \sum_{\ell_i\in\mathcal{L}} \hat{\nu}^*_{\ell_i} \sum_{j\in\mathcal{J}} \hat{Q}^{(n)}_{\ell_i,j}(t). \tag{3.29}$$

where $\hat{Q}^{(n)}_{\ell_i,j}(t)$ is defined in the same fashion as in (3.14).

Since each type i arrival follows a Poisson process, each subtype ℓ_i arrival process remains a Poisson process when MARO−flex is applied. Thus, applying Theorem 2.3.3, we have

$$\hat{Y}^{(n)}_F(t) \xrightarrow{w} \text{RBM}(\theta, \sigma^2), \quad \text{as } n \to \infty,$$

where

$$\theta = \sum_{\ell_i\in\mathcal{L}} \hat{\nu}^*_{\ell_i} \hat{b}_{\ell_i}, \tag{3.30}$$

$$\sigma^2 = \sum_{\ell_i\in\mathcal{L}} (\hat{\nu}^*_{\ell_i})^2 \left[\lambda_{\ell_i} + \sum_{j\in\mathcal{J}} \phi_{\ell_i,j}\mu^3_{i,j}\beta^2_{i,j}\right]. \tag{3.31}$$

The quantity $\hat{\nu}^*_{\ell_i}$ can obtained by solving the LP (3.10) (where ν_i is changed to $\hat{\nu}_\ell$). Since $\hat{\lambda}$ satisfies the CRP condition, we have $\hat{\nu}^*_{\ell_i}\mu_{i,j} = \xi^*_j$ if $\phi_{\ell_i,j} \neq 0$. This implies

$$\hat{\nu}^*_{\ell_i} = \nu^*_i, \tag{3.32}$$

for each group i. Thus, $\hat{Y}^{(n)}_F(t)$ in (3.29) is equivalent to $\hat{Y}^{(n)}_f(t)$ in (3.20).

From (3.32) and (3.28), (3.30) is equivalent to

$$\theta = \sum_{i\in\mathcal{I}} \nu^*_i b_i.$$

The terms in (3.31) which are indexed by the subscript ℓ_i can be grouped with respect to each group i, so that $\sigma^2 = \sum_{i\in\mathcal{I}} \sigma^2_i$. Let $\sum_{\ell_i}$ denote the summation of terms within each group i. We have

$$\sigma^2_i = (\hat{\nu}^*_{\ell_i})^2 \left[\sum_{\ell_i} \lambda_{\ell_i} + \sum_{j\in\mathcal{J}} \sum_{\ell_i} \phi_{\ell_i,j}\mu^3_{i,j}\beta^2_{i,j} \right].$$

Thus, using (3.32) and (3.26), (3.31) is equivalent to

$$\sigma^2 = \sum_{i\in\mathcal{I}} (\nu^*_i)^2 \left[\lambda_i + \sum_{j\in\mathcal{J}} \psi^*_{i,j}\mu^3_{i,j}\beta^2_{i,j} \right].$$

This completes the proof.

□

3.3.4 MARO−tree

In the MARO−flex policy, a type i flexible arrival can choose any one of the queues in the candidate set S_i. If the number of servers J is large, the size of S_i may be large. However, in the circumstance when the choice may be severely limited due to locality constraints or personal preferences (e.g., a patient can only afford to go to hospitals within a certain distance from their home), a new routing policy is needed to address these concerns. So we propose the MARO−tree policy as follows.

Policy 3.3.4 (MARO−tree). *Given $0 < q_i \leq 1$, for each task type i, $100(1 - q_i)$ percent are assumed to be dedicated arrivals; the remaining $100q_i$ percent*

are assumed to be flexible arrivals. Given the set of candidate queues defined in (3.21)*, a dedicated arrival is routed to a queue* j_k *(*$k \in \{1, ..., |S_i|\}$*) with probability*

$$p^d_{i,j_k} = \psi^*_{i,j_k}\mu_{i,j_k}/\tilde{r}_i, \tag{3.33}$$

which is the same as (3.18) *in the MARO–flex policy.*

A flexible arrival picks queue j_k *(*$k \in \{1, ..., |S_i| - 1\}$*) with probability* p^f_{i,j_k} *and joins one of two* adjacent *queues* (j_k, j_{k+1}) *satisfying*

$$j \in \arg \min_{j \in \{j_k, j_{k+1}\}} \frac{C_j'(Z_j)}{\mu_{i,j}}.$$

For each task type i*, if* $|S_i| > 2$*, the routing probabilities* p^f_{i,j_k} *should satisfy the conditions*

$$\sum_{n=1}^{k} p^d_{i,j_n} < \sum_{n=1}^{k} p^f_{i,j_n} < \sum_{n=1}^{k+1} p^d_{i,j_n}, \quad k \in \{1, ..., |S_i| - 2\}, \tag{3.34}$$

and

$$\sum_{k=1}^{|S_i|-1} p^f_{i,j_k} = 1. \tag{3.35}$$

If $|S_i| \leq 2$*, we naturally have* $p^f_{i,j_1} = 1$.

For any type i flexible arrivals choosing to join one of the two adjacent queues (j_k, j_{k+1}), we denote them as subtype i_k. Let $r_{i,j_k} = (\phi_{i_k,j_k}\mu_{i,j_k})$ denote the average rate at which these arrivals join queue j_k. In the proof of Theorem 3.3.3, we will see that the first inequality of (3.34) implies that for the righthand-side queue $r_{i,j_{k+1}} > 0$, while the second inequality of (3.34) implies for the lefthand-side queue $r_{i,j_k} > 0$. Thus the queues in set S_i are able to communicate with each other and the workload may be shifted between the queues.

The sets of queues which type i tasks actually join are S_i for the dedicated arrivals and

$$S^f_i = \begin{cases} \{j_1\}, & \text{if } |S_i| = 1, \\ \{(j_k, j_{k+1}) : j_k, j_{k+1} \in S_i\}, & \text{if } 2 \leq |S_i| \leq J, \end{cases} \tag{3.36}$$

for the flexible arrivals, respectively. Also we have $|S_i^f| = |S_i| - 1$, if $2 \leq |S_i| \leq J$.

Let $K_i = |S_i|$. The average number of servers from which MARO−tree acquires state information is

$$N_{s_{(tree)}} = \sum_{i:K_i>1} \frac{q_i\lambda_i}{\tilde{\lambda}} \sum_{k=1}^{K_i-1} 2p_{i,j_k}^f = 2 \sum_{i:K_i>1} \frac{q_i\lambda_i}{\tilde{\lambda}}.$$

To quantify the reduction in the amount of state information used by the MARO−tree policy, we define the discount

$$Discnt_{(tree)} = \left(1 - \frac{N_{s_{(tree)}}}{J}\right) \times 100\%. \tag{3.37}$$

Similar to $Discnt_{(2k)}$, $Discnt_{(tree)}$ increases independently of the structure of Ψ^* when the number of servers grows.

Heavy Traffic Analysis

We use Theorem 3.3.3 to show that in heavy traffic, the performance of MARO−tree is close to that of MARO−flex. When MARO−tree is operating on the same sequence of systems considered in Theorem 3.3.1, we denote the diffusion scaled total workload process by $\hat{Y}_t^{(n)}(t)$, which is defined in the same fashion as in (3.20). Then we have

Theorem 3.3.3. *$\hat{Y}_t^{(n)}(t) \xrightarrow{w} \hat{Y}_d$, where the diffusion limit is independent of both q_i, the proportion of type i flexible arrivals, and p_{i,j_k}^f, the routing probabilities of the arrivals.*

Proof.

The proof is similar to that of Theorem 3.3.2 except for the parts dealing with the flexible arrivals.

Define the index set $\hat{\mathcal{I}} = \bigcup_{i\in\mathcal{I}} \hat{S}_i^d \cup \bigcup_{i\in\mathcal{I}} \hat{S}_i^f$, where $\hat{S}_i^d = \{(i, j_k) : j_k \in S_i\}$ and $\hat{S}_i^f = \{(i, j_k, j_{k+1}) : (j_k, j_{k+1}) \in S_i^f\}$. Define the set $\mathcal{L} = \{1, ..., |\hat{\mathcal{I}}|\}$ and a one-to-one function $f_2 : \hat{\mathcal{I}} \to \mathcal{L}$ of the form

$$\begin{cases} f_2(1, j_k) & = k, \qquad k \in \{1, ..., K_1\}; \\ f_2(1, j_k, j_{k+1}) & = K_1 + k, \qquad k \in \{1, ..., K_1 - 1\}, \end{cases}$$

and

$$\begin{cases} f_2(i,j_k) & = \sum_{n=1}^{i-1}(2K_n-1)+k, & i>1, k\in\{1,...,K_i\}; \\ f_2(i,j_k,j_{k+1}) & = \sum_{n=1}^{i-1}(2K_n-1)+K_i+k, & i>1, k\in\{1,...,K_i-1\}. \end{cases}$$

The arrival rate vector $\hat{\lambda}=(\hat{\lambda}_\ell)_{1\times|\hat{\mathcal{I}}|}$ has elements grouped with respect to each task type i and indexed by subscripts ℓ_i. In each group i, the arrival rates are

$$\hat{\lambda}_{\ell_i}=\begin{cases}(1-q_i)\lambda_i p^d_{i,j_k}, & \text{for the dedicated arrivals,}\\ q_i\lambda_i p^f_{i,j_k}, & \text{for the flexible arrivals,}\end{cases} \tag{3.38}$$

where the indices ℓ_i are determined by

$$\ell_i=f_2(i,j_k):=\begin{cases} i^d_k, & (i,j_k)\in\hat{S}^d_i,\\ i^f_k, & (i,j_k,j_{k+1})\in\hat{S}^f_i.\end{cases} \tag{3.39}$$

So for each type i, we have

$$\begin{cases}\sum_{k=1}^{|S_i|}\hat{\lambda}_{i^d_k}=(1-q_i)\lambda_i,\\ \sum_{k=1}^{|S^f_i|}\hat{\lambda}_{i^f_k}=q_i\lambda_i.\end{cases}$$

Similar to (3.25), for each task type i ($i\in\mathcal{I}$), the routing structure matrix (with only non-zero elements shown) is

$$\Phi_i=\left[\frac{\Phi^d_i}{\Phi^f_i}\right]=\left[\begin{array}{ccccccc} \phi_{i^d_1,j_1} & & & & & & \\ & \ddots & & & & & \\ & & & & & & \phi_{i^d_{K_i},j_{K_i}} \\ \hline \phi_{i^f_1,j_1} & \phi_{i^f_1,j_2} & & & & & \\ & \phi_{i^f_2,j_2} & \phi_{i^f_2,j_3} & & & & \\ & & \ddots & & \ddots & & \\ & & & & & \phi_{i^f_{K_i-1},j_{K_i-1}} & \phi_{i^f_{K_i-1},j_{K_i}} \end{array}\right]_{(2K_i-1)\times J} \tag{3.40}$$

where the row indices i^d_k and i^f_k (denoted collectively as ℓ_i) are determined

using (3.39). Let the non-zero elements ϕ_{ℓ_i,j_k} be

$$\begin{cases} \phi_{i_k^d,j_k} & = (1-q_i)\psi^*_{i,j_k}, & k \in \{1,...,K_i\}, \\ \phi_{i_1^f,j_1} & = q_i\psi^*_{i,j_1}, & \\ \phi_{i_{k-1}^f,j_k} & = \frac{q_i}{\mu_{i,j_k}}\left(\lambda_i \sum_{n=1}^{k-1} p^f_{i,j_n} - \sum_{n=1}^{k-1}\psi^*_{i,j_n}\mu_{i,j_n}\right), & k \in \{2,...,K_i-1\}, \\ \phi_{i_k^f,j_k} & = q_i\psi^*_{i,j_k} - \phi_{i_{k-1}^f,j_k}, & k \in \{2,...,K_i-1\}, \\ \phi_{i_{K_i-1}^f,j_{K_i}} & = q_i\psi^*_{i,j_{K_i}}. & \end{cases} \tag{3.41}$$

Given $0 < p_i \leq 1$ and condition (3.34), all the elements ϕ_{ℓ_i,j_k} are positive for the flexible arrivals. This combined with Assumption 3.3.3 (the matrix Ψ^* representing a tree), implies that the entire routing structure matrix $\Phi = (\Phi_i)_{I\times 1}$ also represents a tree, with nodes being arrival subtypes ℓ and queues j and arcs (ℓ, j) being the actual routing activities.

From (3.33), (3.38) and (3.41), we have for each task type i a set of equalities

$$\begin{cases} \phi_{i_k^d,j_k}\mu_{i,j_k} & = \hat{\lambda}_{i_k^d}, & k \in \{1,...,K_i\}, \\ \phi_{i_k^f,j_k}\mu_{i,j_k} + \phi_{i_k^f,j_{k+1}}\mu_{i,j_{k+1}} & = \hat{\lambda}_{i_k^f}, & k \in \{1,...,K_i-1\}, \\ \sum_{\ell_i}\phi_{\ell_i,j_k} & = \psi^*_{i,j_k}, & k \in \{1,...,K_i\}. \end{cases} \tag{3.42}$$

Assumption 3.3.2 and (3.42) imply that $\phi_{\ell,j}$ defined in (3.41) is a unique solution to the linear system (3.27). Therefore, from Theorem 2.3.1, the CRP condition holds for the arrival rate vector $\hat{\lambda}$ defined in (3.38).

For each task type i, let

$$\hat{b}_{\ell_i} = \begin{cases} (1-q_i)b_i p^d_{i,j_k}, & \text{for the dedicated arrivals,} \\ q_i b_i p^f_{i,j_k}, & \text{for the flexible arrivals,} \end{cases} \tag{3.43}$$

so that $\sum_{\ell_i}\hat{b}_{\ell_i} = b_i$. From (3.13) and (3.43), we have

$$\lim_{n\to\infty}\sqrt{n}\left(\hat{\lambda}^{(n)}_{\ell_i} - \hat{\lambda}_{\ell_i}\right) = \hat{b}_{\ell_i},$$

where $\hat{\lambda}_{\ell_i}$ is defined in (3.38). Then using the same reasoning as in the proof of Theorem 3.3.2, we have the result.

□

Routing Probabilities of the Flexible Arrivals

We give in (3.34) and (3.35) the conditions on the routing probabilities, p^f_{i,j_k} ($k \in \{1,...,K_i-1\}$, $K_i = |S_i| > 2$), with which a flexible arrival picks a pair of adjacent queues. Here we discuss a method of explicitly choosing the probabilities.

The proposed means of choosing p^f_{i,j_k} aims both to maximize the amount of (flexible) workload that can be shifted between servers and to balance the loads among servers. The reasons are (1) from the analysis of MARO's properties (in Section 3.3.1), we see that the dynamic routing policy benefits from doing short term shifting of workload between servers to avoid congestion; (2) the long-run server utilizations, on the other hand, also have a big impact on the system performance.

In the proof of Theorem 3.3.3, we know that given the matrix Φ_i in (3.40), the rate at which type i flexible arrivals join queue j_k is

$$\hat{\lambda}_{j_k^f} = \begin{cases} \phi_{i^f_1,j_1}\mu_{i,j_1}, & k=1, \\ \phi_{i^f_{k-1},j_k}\mu_{i,j_k} + \phi_{i^f_k,j_k}\mu_{i,j_k}, & k \in \{2,...,K_i-1\}, \\ \phi_{i^f_{K_i-1},j_{K_i}}\mu_{i,j_{K_i}}, & k=K_i. \end{cases} \tag{3.44}$$

On the other hand, the rate at which the dedicated arrivals join queue j_k is $\hat{\lambda}_{i^d_k}$ given in (3.38). Therefore, p^f_{i,j_k} is chosen to maximize the difference between the two rates $(\hat{\lambda}_{j_k^f} - \hat{\lambda}_{i^d_k})$ for all $j_k \in S_i$. This involves solving a linear programming problem as follows.

For each task type i, define the matrix

$$P_i^f = \begin{bmatrix} \varphi_{1,1} & \varphi_{1,2} & & & \\ & \varphi_{2,2} & \varphi_{2,3} & & \\ & & \ddots & \ddots & \\ & & & \varphi_{K_i-1,K_i-1} & \varphi_{K_i-1,K_i} \end{bmatrix}_{(K_i-1)\times K_i}, \tag{3.45}$$

where the non-zero elements are

$$\varphi_{k,k} = \phi_{i^f_k,j_k}\mu_{i,j_k}/(q_i\lambda_i) \quad \text{and} \quad \varphi_{k,k+1} = \phi_{i^f_k,j_{k+1}}\mu_{i,j_{k+1}}/(q_i\lambda_i),$$

for $k \in \{1, ..., K_i - 1\}$. The LP is hence given by

$$\max_{(\gamma_i, P_i^f)} \quad \gamma_i \tag{3.46}$$

$$\text{s.t.} \quad q_i\varphi_{1,1} - (1-q_i)p^d_{i,j_1} \geq \gamma_i, \tag{3.47}$$

$$q_i(\varphi_{k,k} + \varphi_{k-1,k}) - (1-q_i)p^d_{i,j_k} \geq \gamma_i, \forall k \in \{2, ..., K_i - 1\}, \tag{3.48}$$

$$q_i\varphi_{K_i-1,K_i} - (1-q_i)p^d_{i,j_{K_i}} \geq \gamma_i, \tag{3.49}$$

$$\sum_{n=1}^{k} \varphi_{n,n} + \varphi_{n,n+1} > \sum_{n=1}^{k} p^d_{i,j_n}, \quad \forall k \in \{1, ..., K_i - 2\}, \tag{3.50}$$

$$\sum_{n=1}^{k} \varphi_{n,n} + \varphi_{n,n+1} < \sum_{n=1}^{k+1} p^d_{i,j_n}, \quad \forall k \in \{1, ..., K_i - 2\}, \tag{3.51}$$

$$\sum_{\ell=1}^{K_i-1} \sum_{k=1}^{K_i} \varphi_{\ell,k} = 1, \tag{3.52}$$

$$\varphi_{\ell,k} > 0, \quad \forall \ell, \forall k, \tag{3.53}$$

where p^d_{i,j_k} is given in (3.33) and q_i is the proportion of flexible arrivals. Constraints (3.47)–(3.49) are derived from (3.38), (3.42) and (3.44). Constraints (3.50)–(3.52) are equivalent to conditions (3.34) and (3.35), since from (3.38) and (3.42), we have

$$p^f_{i,j_k} = \varphi_{k,k} + \varphi_{k,k+1}. \tag{3.54}$$

To transform the strict inequalities to the standard form for LP solvers, a small number $\epsilon > 0$ is added to the left-hand sides of (3.50) and (3.53); similarly $-\epsilon < 0$ is added to (3.51).

When q_i and p^d_{i,j_k} are such that constraints (3.47)–(3.49) are tight at the optimum, each column k of P_i^f sums to $\left[\gamma_i^* + (1-q_i)p^d_{i,j_k}\right]/q_i$, where $\gamma_i^* = (2q_i - 1)/K_i$, $k \in \{1, ..., K_i\}$, and each row of P_i^f sums to p^f_{i,j_k}, $k \in \{1, ..., K_i - 1\}$.

Here we give an example to show the actual utilizations of the servers in a heterogeneous system using LP (3.46). This example is abstracted from the Grid application in Section 5.2.1.

Example 3.3.3. *Let $I = 2$, $J = 6$ and $q_i = 1$ for each task type i. The first-order primitives are*

$$\lambda = \begin{bmatrix} 151 & 50.3 \end{bmatrix},$$

and

$$\mu = \begin{bmatrix} 17 & 25 & 24 & 29 & 48 & 26 \\ 30 & 48 & 78 & 84 & 145 & 136 \end{bmatrix}.$$

Then LP (3.2) *has a solution* $\rho^* = 0.95$ *and*

$$\Psi^* = \begin{bmatrix} 0.950 & 0.950 & 0.950 & 0.950 & 0.950 & 0.580 \\ & & & & & 0.370 \end{bmatrix}.$$

For the dedicated arrivals, the routing probability matrix is

$$P^d = (p^d_{i,j})_{I\times J} = \begin{bmatrix} 0.107 & 0.157 & 0.151 & 0.183 & 0.302 & 0.100 \\ & & & & & 1.000 \end{bmatrix}.$$

Assuming $q_i = 1$ *for all task types* i*, the solution of LP* (3.46) *for type 1 tasks is* $\gamma^* = 1/6$ *and*

$$P_1^{f*} = \begin{bmatrix} 0.167 & 0.052 & & & & \\ & 0.115 & 0.035 & & & \\ & & 0.132 & 0.040 & & \\ & & & 0.127 & 0.080 & \\ & & & & 0.087 & 0.167 \end{bmatrix}$$

for task type 1. Each column of P_1^{f*} *sums to* γ^**, which means constraints* (3.47)–(3.49) *are tight at the optimum.*

Summing each row of P_1^{f*}*, we have for type 1 flexible tasks the routing probabilities*

$$(p^f_{1,j})_{1\times(J-1)} = \begin{bmatrix} 0.219 & 0.150 & 0.172 & 0.207 & 0.254 \end{bmatrix}.$$

Given P^d*, all type 2 flexible tasks are routed to queue 5 and join one of the two queues 5 or 6, according to which has a shorter expected waiting time.*

Simulation yields the corresponding steady state total mean queue length of the system to be $38.94 \pm 2.3\%$ *at a* 95% *confidence level. The utilizations of the servers are given by the vector*

$$(\rho_j)_{1\times J} = \begin{bmatrix} 0.91 & 0.98 & 0.95 & 0.95 & 0.96 & 0.94 \end{bmatrix}.$$

The average and standard deviation of the values ρ_j *are* 0.95 *and* 0.02*, respectively. The result fits well to the solution of the allocation LP* (3.2)*, indicating that systems using the dynamic routing policy MARO–tree can achieve the optimal objective of LP* (3.2).

Ring Routing Structure

Motivated by Theorem 3.3.3, if the mechanism for good performance is that a sufficient proportion of incoming workload can be shifted from any server to any other server through the routing structure, then there are two natural choices that should intuitively lead to better performance (while still keeping the number of choices to be at most two). One of these is the MARO$-2/k$ policy described in Section 3.3.2, as it can spread incoming workload over many different queues, so it seems reasonable that it would have better performance. Another obvious choice is to extend the MARO$-$tree policy so that there is an additional stream of flexible arrivals that is allowed to join the shorter of queues J and 1. This would allow incoming workload to be shifted bidirectionally, rather than unidirectionally for the two end queues. We will call such a routing structure a "ring" structure, as opposed to the "tree" structure of our original policy. Unfortunately, as seen below, the CRP condition does not hold for either of these, but we suggest a means to make a comparison.

First, we assume all arrivals flexible for the ring structure, since the diffusion limit in Theorem 3.3.3 is independent of the flexibility level q. The ring structure allows a type i arrival to pick queue j_k ($k \in \{1, ..., |S_i|\}$) with probability p^f_{i,j_k} and joins one of two queues $(j_k, j_{k+1\,(\mathrm{mod}\ K_i)})$ satisfying

$$j \in \arg \min_{j \in \{j_k, j_{k+1\,(\mathrm{mod}\ K_i)}\}} \frac{C_j'(Z_j)}{\mu_{i,j}},$$

where $K_i = |S_i|$ and $\sum_{k=1}^{K_i} p^f_{i,j_k} = 1$. Therefore, the sets of queues which type i tasks actually join is

$$S_i^f = \begin{cases} \{j_1\}, & \text{if } |S_i| = 1, \\ \left\{(j_k, j_{k+1\,(\mathrm{mod}\ K_i)}) : j_k, j_{k+1\,(\mathrm{mod}\ K_i)} \in S_i\right\}, & \text{if } 2 \le |S_i| \le J, \end{cases}$$

for the flexible arrivals. Also we have $|S_i^f| = K_i$.

Define the index sets $\hat{\mathcal{I}} = \bigcup_{i \in \mathcal{I}} \hat{S}_i^f$, where $\hat{S}_i^f = \{(i, j_k, j_{k+1\,(\mathrm{mod}\ K_i)}) : (j_k, j_{k+1\,(\mathrm{mod}\ K_i)}) \in S_i^f\}$ and $\mathcal{L} = \{1, ..., |\hat{\mathcal{I}}|\}$. Define a one-to-one function $f_3 : \hat{\mathcal{I}} \to \mathcal{L}$ of the form

$$\begin{cases} f_3(1, j_k, j_{k+1\,(\mathrm{mod}\ K_i)}) &= k, & k \in \{1, ..., K_1\}; \\ f_3(i, j_k, j_{k+1\,(\mathrm{mod}\ K_i)}) &= \sum_{n=1}^{i-1} K_n + k, & i > 1, k \in \{1, ..., K_i\}. \end{cases}$$

The arrival rate vector $\hat{\lambda} = (\hat{\lambda}_\ell)_{1\times|\hat{\mathcal{I}}|}$ has elements grouped with respect to each task type i and indexed by subscripts ℓ_i. In each group i, the arrival rates are

$$\hat{\lambda}_{\ell_i} = \lambda_i p^f_{i,j_k}, \tag{3.55}$$

where the indices ℓ_i are determined by

$$\ell_i = f_3(i, j_k) := i^f_k, \qquad (i, j_k, j_{k+1\,(\mathrm{mod}\ K_i)}) \in \hat{S}^f_i.$$

So for each type i, we have $\sum_{k=1}^{K_i} \hat{\lambda}_{i^f_k} = \lambda_i$.

The difficulty in analyzing the ring structure is that the corresponding arrival rate vector $\hat{\lambda}$ defined by (3.55) does not satisfy the CRP condition, because for each task type i, the routing structure matrix

$$\Phi_i = \begin{bmatrix} \phi_{i^f_1,j_1} & \phi_{i^f_1,j_2} & & & & \\ & \phi_{i^f_2,j_2} & \phi_{i^f_2,j_3} & & & \\ & & \ddots & \ddots & & \\ & & & \phi_{i^f_{K_i-1},j_{K_i-1}} & \phi_{i^f_{K_i-1},j_{K_i}} & \\ \phi_{i^f_{K_i},j_1} & & & & \phi_{i^f_{K_i},j_{K_i}} & \end{bmatrix}_{K_i\times J} \tag{3.56}$$

has a cycle in the corresponding graph. That implies the entire routing structure matrix

$$\Phi = (\Phi_i)_{I\times 1} = (\phi_{\ell,j})_{|\hat{\mathcal{I}}|\times J}$$

has rings and there are multiple solutions to the linear system (3.27). Therefore, by Theorem 2.3.1, the CRP condition does not hold and we cannot directly make conclusions similar to Theorem 3.3.3.

However, this does not imply that the ring routing structure will have performance worse than the tree routing structure. To see this, we modify the ring structure so that the flexible arrivals which join the shorter of the two end queues j_1 and j_{K_i} instead join each of those two queues with equal probability. Then the corresponding routing structure matrix has the same form as (3.40) for each task type i, since the last row in (3.56) becomes the dedicated part Φ^d_i in (3.40). Using the same reasoning in the proof of Theorem 3.3.3, we conclude that the system in which the modified version of the ring routing structure is

applied, has the total workload process achieving the same RBM limit as that of the original tree model.

At this point, we suggest that the modification that has been made would degrade the performance in the sense that if L_R is the mean number in the system for the ring routing structure and $\tilde{L}_R$ is the mean number in the system for the modified ring structure, then $L_R \leq \tilde{L}_R$. While we are unable to provide a proof, the intuition is that if queue j_1 has a much higher workload than queue j_{K_i}, the original ring structure enables the incoming workload to be shifted directly to queue j_{K_i}, while the modified structure only allows sequential shifting through queues j_2 to j_{K_i-1}. So congestion is alleviated more quickly in the original ring structure. Similar reasoning implies that the MARO$-2/k$ policy should perform better than MARO$-$tree.

3.3.5 Comparison

Here we compare MARO with its variants, in terms of the discount of the average required state information for making routing decisions. To compare the discounted amounts, we use (3.8), (3.16), (3.19) and (3.37), which correspond to MARO, MARO$-2/k$, MARO$-$flex and MARO$-$tree, respectively.

Suppose (1) the number of candidate queues $|S_i|$ is greater than 1 for each task type i, (2) $\sum_{i\in\mathcal{I}} |S_i| = I + J - 1$ and (3) the proportion of flexible arrivals q_i equals a constant q for all $i \in \mathcal{I}$. A quantitative comparison is given in two cases: the equal-arrival-rate case and the single-dominant-arrival case.

Equal-Arrival-Rate Case

Let $p_i^t = \lambda_i/\tilde{\lambda} = 1/I$ for all $i \in \mathcal{I}$, i.e., all arrival rates are equal. We have

$$\begin{aligned} Discnt_{(MARO)} &= [(I-1)(J-1)]/(IJ) \times 100\%, \\ Discnt_{(2k)} &= (J-2)/J \times 100\%, \\ Discnt_{(flx)} &= [(I-q)J - q(I-1)]/(IJ) \times 100\%, \\ Discnt_{(tree)} &= (J-2q)/J \times 100\%. \end{aligned}$$

It can be shown that in this case, all the *Discnt* values are monotonically increasing in the number of servers J, when the number of task types I is

greater than one. Additionally, we have $Discnt_{(tree)} \geq Discnt_{(flx)}$ if $J \geq I+1$.

Figure 3.1 shows the comparison in the equal-arrival-rate case, given the number of task types $I = 4$. The values of $Discnt_{(MARO)}$ and $Discnt_{(2k)}$ are the same as those of $Discnt_{(flx)}$ and $Discnt_{(tree)}$ with $q = 1$, respectively. We can see that for the same policy, less state information is required as the proportion of flexible arrivals increases. At the same level of flexibility q, MARO−tree requires less state information than MARO−flex.

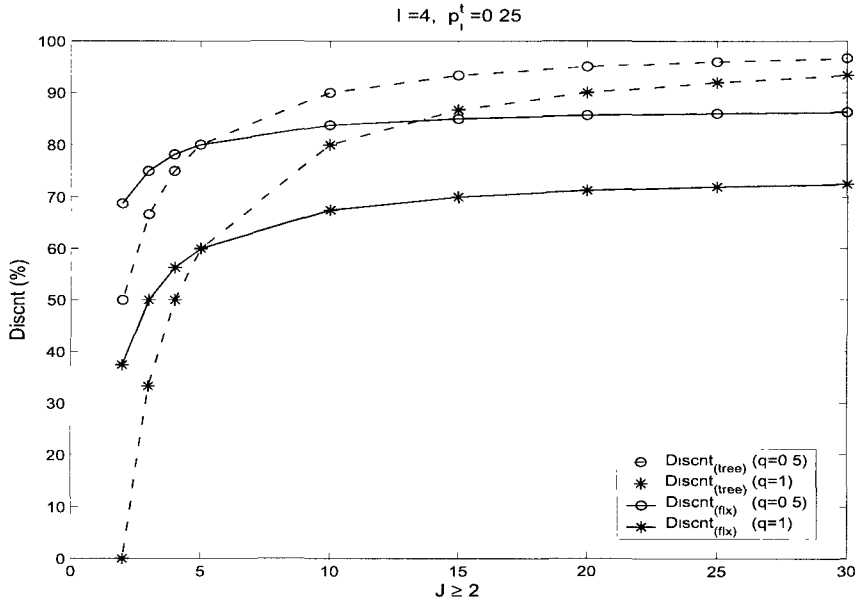

Figure 3.1: *Discnt* vs. number of servers J, equal-arrival-case case

Single-Dominant-Arrival Case

Let $p_1^t = \lambda_1/\tilde{\lambda} > 1/I$ and $p_i^t = \lambda_i/\tilde{\lambda} = (1-p_1^t)/(I-1)$ for all $i \in \mathcal{I} \setminus \{1\}$. If p_1^t is close to one, type 1 tasks are dominant. Also let $|S_1| = J + 1 - I \geq 2$ and $|S_i| = 2$ for all $i \in \mathcal{I} \setminus \{1\}$, i.e., the dominant task type has to acquire state information from the largest possible set of candidate queues (which might be considered as a worst case). Then $Discnt_{(2k)}$ and $Discnt_{(tree)}$ are the same as in the equal-arrival-rate case. In addition, we have

$$Discnt_{(flx)} = \left[(1-qp_1^t)J - q(2-p_1^t-p_1^t I)\right]/J \times 100\%.$$

It can be seen that (1) if $p_1^t > 2/(I+1)$, $Discnt_{(flx)}$ (including $Discnt_{(MARO)}$) is monotonically decreasing in the number of servers J and bounded below by $(1 - qp_1^t)$, as $J \to \infty$. Since $J \geq I + 1$ in this case, we have $Discnt_{(tree)} \geq Discnt_{(flx)}$. Figure 3.2 shows the comparison in two single-dominant-arrival cases, given $I = 4$ and p_1^t is equal to 0.9 and 0.5, respectively; (2) if $1/I < p_1^t < 2/(I+1)$, $Discnt_{(flx)}$ is monotonically increasing in the number of servers J. However, given I and J, $Discnt_{(flx)}$ is still smaller than that in the equal-arrival-rate case, regardless of the flexibility level q.

3.4 Special Cases

In this section, we discuss several cases where the MARO related policies are applied to systems with homogeneous servers, i.e., the processing time distribution of any task type i is the same at each server j. Referring to the matrices μ and β defined in Section 3.1, we have $\mu_{i,j} = \mu_i$ and $\beta_{i,j} = \beta_i$ for all $j \in \mathcal{J}$. In Section 3.4.1, we give the special forms that the MARO related policies will take when there is only one task type. The results are extended to multiple task types in Section 3.4.2.

As an additional special case, we discuss in Section 3.4.3 how the MARO related policies are applied to a system with a single task type and heterogeneous servers. For studies of systems with multiple task types and heterogeneous servers, we present results in Chapter 5.

3.4.1 Homogeneous Systems with Single Task Type

In this case, the first-order primitives are a scalar λ_1 and a vector $\mu = (\mu_1)_{1\times J}$. The solution to LP (3.2) is

$$\rho^* = \lambda_1/(J\mu_1), \quad \Psi^* = (\rho^*)_{1\times J}.$$

Hence, the set of candidate queues is $S_1 = \mathcal{J}$.

The MARO policy is the same as MinDrift(Q) (Policy 3.2.1), which is equivalent to routing a task to queue j satisfying

$$j \in \arg\min_{j\in\mathcal{J}} C_j'(Z_j).$$

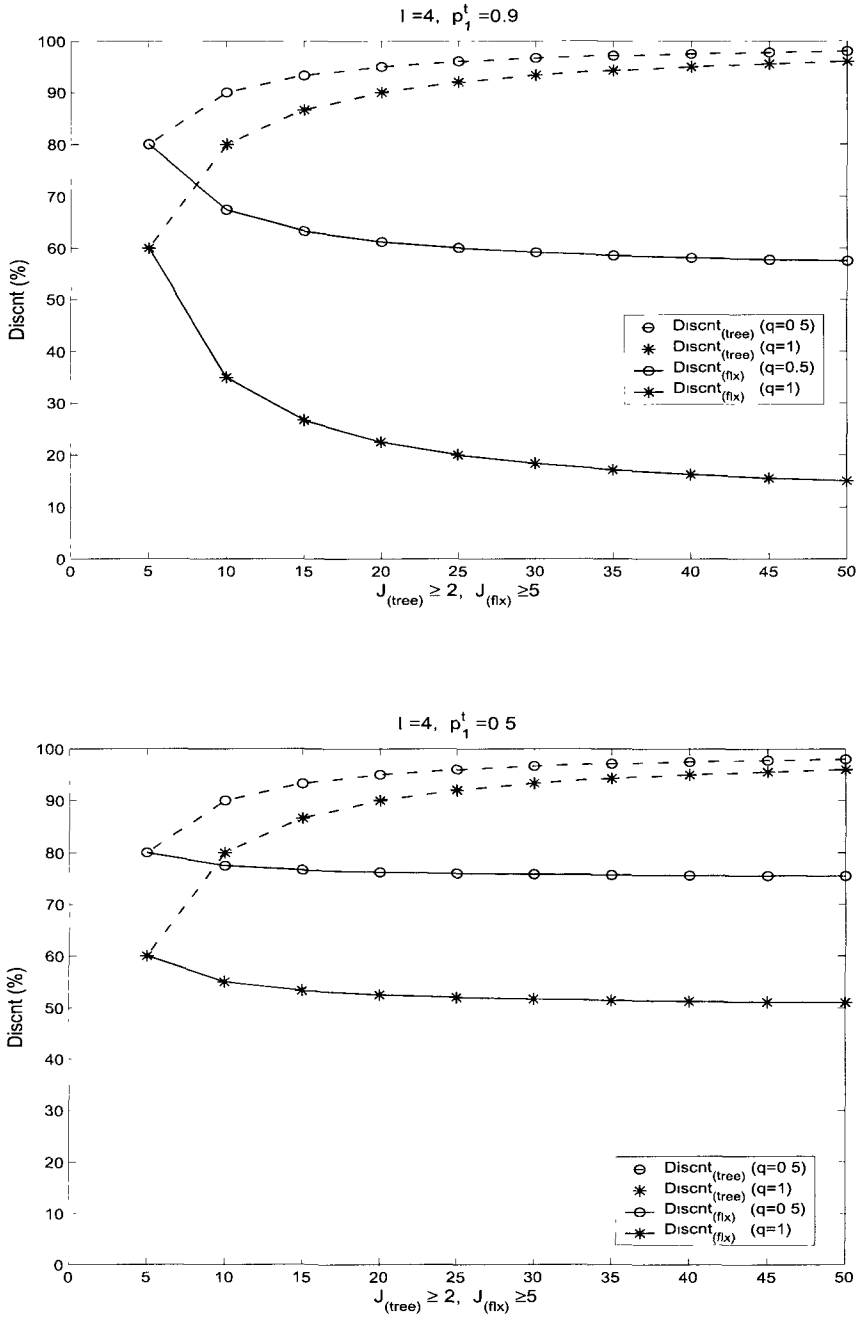

Figure 3.2: *Discnt* vs. number of servers J, single-dominant-arrival case

If the cost function is of the form $C_j(Z_j) = cZ_j^2$, where $c > 0$ is a constant and $Z_j(t)$ is the expected waiting time at each arrival time t given by

$$Z_j(t) = Q_j(t)/\mu_1,$$

then MARO (and MARO−flex as well) is the same as routing a (flexible) task to queue j with the shortest queue length $Q_j(t)$, i.e., the "join the shortest queue" (JSQ) policy.

The MARO−$2/k$ and MARO−tree policies are changed accordingly: a flexible arrival is routed to the shorter one of the paired queues. Using MARO−$2/k$, the two candidate queues are picked with probabilities $1/J$ and $1/(J-1)$, respectively. In the MARO−tree policy, dedicated arrivals are routed to each queue j ($j \in \mathcal{J}$) with equal probability $p_j^d = 1/J$. Flexible arrivals are routed to queue j with probability p_j^f, which can be determined in the following way as one solution to LP (3.46).

Let $q \in (0, 1]$ be the proportion of flexible arrivals. Since constraints (3.47)–(3.49) are tight at optima, the optimal solution is

$$\gamma^* = (2q-1)/J$$

and

$$P^{f*} = \begin{bmatrix} \frac{1}{J} & \frac{1}{J} - \varphi_{2,2}^* & & & \\ & \varphi_{2,2}^* & \frac{1}{J} - \varphi_{3,3}^* & & \\ & & \ddots & \ddots & \\ & & & \varphi_{J-1,J-1}^* & \frac{1}{J} \end{bmatrix}_{(J-1)\times J}.$$

Let $\varphi_{j,j}^* = \varphi_{j-1,j}^* = 1/(2J)$ for $j \in \{2, ..., J-1\}$. We have

$$p_j^f = \begin{cases} 3/(2J), & j \in \{1, J-1\}, \\ 1/J, & j \in \{2, ..., J-2\}, \end{cases} \tag{3.57}$$

which satisfy conditions (3.34) and (3.35).

Finally, we compare the discounts of the average required state information for the MARO related policies, when they are applied to systems with

a single task type. From (3.8), (3.16), (3.19) and (3.37), we have

$$\begin{aligned} Discnt_{(MARO)} &= 0\%, \\ Discnt_{(2k)} &= (J-2)/J \times 100\%, \\ Discnt_{(flx)} &= (1-q) \times 100\%, \\ Discnt_{(tree)} &= (J-2q)/J \times 100\%. \end{aligned} \tag{3.58}$$

Since $J \geq 2$ and $q \leq 1$, we have $Discnt_{(tree)} \geq Discnt_{(flx)} \geq Discnt_{(MARO)}$ and $Discnt_{(tree)} \geq Discnt_{(2k)}$. That is MARO−tree requires the least amount of information to make routing decisions.

Heavy Traffic Analysis

In the case of a single task type, Assumption 3.3.3 is satisfied, since $|S_1| = I + J - 1$. If $\lambda_1 = J\mu_1$ (i.e., Assumption 3.3.2) also holds, then the heavy traffic optimality properties given by Theorems 3.3.1, 3.3.2 and 3.3.3 hold for the MARO, MARO−flex and MARO−tree policies, respectively. Specifically, we have the following results.

Define the total queue length process

$$Q(t) = \sum_{j \in \mathcal{J}} Q_j(t),$$

and its diffusion scaled version

$$\hat{Q}^{(n)}(t) = \frac{1}{\sqrt{n}} Q^{(n)}(nt).$$

Let the total queue length processes for the MARO−flex, MARO−tree and JSQ policies be given by $\hat{Q}^{(n)}_{sf}(t)$, $\hat{Q}^{(n)}_{st}(t)$ and $\hat{Q}^{(n)}_{JSQ}(t)$, respectively. In addition, let $\hat{Q}^{(n)}_{M}(t)$ denote the queue length process for an $M/G/J$ system (i.e., a single queue with J homogeneous servers).

Assume that conditions (3.11) and (3.12) are hold. In addition, the heavy traffic condition

$$\lim_{n \to \infty} \sqrt{n}\left(\lambda_1^{(n)} - \lambda_1\right) = b_1 \tag{3.59}$$

for some finite constant b_1 is assumed to be true. Then we have

Theorem 3.4.1.

(i) For MARO–flex, the diffusion scaled total queue length process, $\hat{Q}_{sf}^{(n)}(t)$, converges weakly to a one-dimensional reflected Brownian motion $\hat{Q}_s$. To be precise, $\hat{Q}_{sf}^{(n)}(t) \xrightarrow{w} \hat{Q}_s = RBM(b_1,\ \lambda_1(1+\mu_1^2\beta_1^2))$, as $n \to \infty$.

(ii) For MARO–tree, $\hat{Q}_{st}^{(n)}(t) \xrightarrow{w} \hat{Q}_s$.

(iii) For JSQ, $\hat{Q}_{JSQ}^{(n)}(t) \xrightarrow{w} \hat{Q}_s$.

(iv) For M/G/J, $\hat{Q}_{M}^{(n)}(t) \xrightarrow{w} \hat{Q}_s$.

Part (iv) of Theorem 3.4.1 implies that when the system is under heavy loads, the MARO related policies should have performance close to that of a system where no routing decision is required, such a system providing a lower bound on achievable performance.

Proof.

(i) From (3.20), we have

$$\hat{Y}_f^{(n)}(t) = \nu_1^* \hat{Q}_{sf}^{(n)}(t),$$

where the arrival workload contribution ν_1^* equals $1/(J\mu)$ in this homogeneous case. Using Theorem 3.3.2, we have

$$\frac{\hat{Q}_{sf}^{(n)}(t)}{J\mu} \xrightarrow{w} \text{RBM}\left(\frac{b_1}{J\mu},\ \frac{\lambda_1(1+\mu_1^2\beta_1^2)}{(J\mu)^2}\right),$$

which yields the result.

(ii) Using Theorem 3.3.3 and reasoning similar to that in part (i), we have the result.

(iii) Since JSQ is equivalent to MARO in this special case, applying Theorem 3.3.1 yields the result. It is noted that by applying Theorem 3.1 from Zhang and Hsu [72], the same result can be obtained.

(iv) A direct application of Theorem 2.2.5 gives the result.

□

3.4.2 Homogeneous Systems with Multiple Task Types

In this case, the first-order primitives are the vector $\lambda = (\lambda_i)_{1\times I}$, $I > 1$ and the matrix μ which consists of J identical column vectors $(\mu_i)_{I\times 1}$, $i \in \mathcal{I}$. Let $\rho_i = \lambda_i/\mu_i$. Applying Corollary 3 in [7], a solution to LP (3.2) is

$$\rho^* = \sum_{i=1}^{I} \rho_i/J$$

and

$$\Psi^* = \begin{bmatrix} \rho_1/J & \cdots & \rho_1/J \\ \rho_2/J & \cdots & \rho_2/J \\ \vdots & & \vdots \\ \rho_I/J & \cdots & \rho_I/J \end{bmatrix}_{I\times J}. \tag{3.60}$$

Therefore, Assumption 3.3.1 holds for this case. The associated set of candidate queues is $S_i = \mathcal{J}$ for all $i \in \mathcal{I}$.

Note that given Proposition 3.3.1, there exists another solution $\tilde{\Psi}^*$ with the maximum number of zero elements. Comparing the number of positive elements N_p, we have

$$\frac{\tilde{N}_p}{N_p} \leq \frac{1}{I} + \frac{1-1/I}{J}.$$

The matrix $\tilde{\Psi}^*$ is sparse when the values of I and J are large. Using $\tilde{\Psi}^*$ in MARO and its variant policies means much less state information is required for making routing decisions. However, it does not necessarily result in better performance of the system. The reason is that in this homogeneous system, using $\tilde{\Psi}^*$ with the maximum number of zero elements limits the choice of candidate servers to shift workload. This can unbalance the workloads between servers to some extent, which yields greater mean queue length of the system (see Chapter 5, Table 5.12).

Using the full matrix Ψ^*, MARO is the same as MinDrift(Q), which is equivalent to routing a type i task to queue j satisfying

$$j \in \arg\min_{j\in\mathcal{J}} \frac{C_j'(Z_j)}{\mu_i}.$$

If the cost function is of the form $C_j(Z_j) = c\mu_i Z_j^2$, where $c > 0$ is a constant and $Z_j(t)$ is the Q-estimated workload at each arrival time t given by

$$Z_j(t) = \sum_{i=1}^{I} Q_{i,j}(t)/\mu_i,$$

then MARO (and MARO−flex as well) is the same as the "join the shortest expected waiting time" (JSEW) policy.

The MARO$-2/k$ and MARO−tree policies are changed accordingly: a flexible arrival is routed to one of the paired queues, according to whichever has a shorter expected waiting time. Using MARO$-2/k$, the two candidate queues are picked with probabilities $1/J$ and $1/(J-1)$, respectively. In the MARO−tree policy, dedicated arrivals are routed to each queue j ($j \in \mathcal{J}$) with equal probability $p_j^d = 1/J$. Flexible arrivals are routed to queue j with probability p_j^f, which can be determined in the same way as (3.57).

Finally, the comparison of the discounted amount of the required state information is the same as that in (3.58) for the MARO related policies.

Heavy Traffic Analysis

Note that if the condition

$$\sum_{i \in \mathcal{I}} \lambda_i/\mu_i = J$$

is true, LP (3.2) has *multiple* solutions (ρ^*, Ψ^*), where $\rho^* = \sum_{i=1}^{I} \psi_{i,j}^* = 1$, for all $j \in \mathcal{J}$. This implies the system is under heavy traffic. However, Assumptions 3.3.2 and 3.3.3 do not hold for the primitives (λ, μ). The complete resource pooling condition thus does not hold for systems which use MinDrift(Q) or the MARO related routing policies. Even with the arrival workload contribution $\nu_i^* = 1/(J\mu_i)$ unique, the MinDrift(Q) policy need not be asymptotically optimal. Roughly speaking, multiple solutions of Ψ^* yield that the RBM variance given in (3.15) is not well defined. However, if a system uses a static routing policy, i.e., type i tasks are routed to queue j with probability $p_{i,j}$ without state information, the diffusion scaled total queue length process still weakly converges to a one-dimensional reflected Brownian motion. We study this case in Chapter 4.

3.4.3 Heterogeneous Systems with Single Task Type

In this case, the first-order primitives are a scalar λ_1 and a vector $\mu = (\mu_j)_{1\times J}$. Let $\tilde{\mu} = \sum_{j=1}^{J} \mu_j$. Applying Corollary 1 in [7], the solution to LP (3.2) is

$$\rho^* = \lambda_1/\tilde{\mu}, \quad \Psi^* = (\rho^*)_{1\times J}.$$

Hence, the set of candidate queues is $S_1 = \mathcal{J}$.

Let the Q-estimated workload at arrival time t be

$$Z_j(t) = Q_j(t)/\mu_j.$$

If the cost function is of the form $C_j(Z_j) = \mu_j Z_j^2$, then MARO and MARO−flex are the same as the JSEW policy which routes a task to queue j satisfying

$$j \in \arg\min_{j\in\mathcal{J}} \frac{Q_j(t)}{\mu_j}.$$

The MARO−$2/k$ and MARO−tree policies are changed accordingly. Using MARO−$2/k$, the two candidate queues $(j,\ j+1)$ are picked with probabilities $\mu_j/\tilde{\mu}$ and $\mu_{j+1}/(\tilde{\mu}-\mu_j)$, respectively. In the MARO−tree policy, dedicated arrivals are routed to queue j ($j \in \mathcal{J}$) with probability $p_j^d = \mu_j/\tilde{\mu}$; flexible arrivals are routed to queue j with probability p_j^f, which is determined using LP (3.46).

Heavy Traffic Analysis

Similar to homogeneous systems with a single task type, we can show that for heterogeneous systems under heavy traffic, the MARO related policies should have performance close to the lower bound on achievable performance.

Let $\hat{Q}_{Mh}^{(n)}(t)$ denote the queue length process for an $M/G/J^h$ system (i.e., a single queue with J heterogeneous servers, service being first-come-first-serve). Assume $\lambda_1 = \tilde{\mu}$ is true, so that Assumption 3.3.2 holds. In addition, conditions (3.11), (3.12) and (3.59) are assumed true. Then we have

Theorem 3.4.2.

(i) For MARO−flex, the diffusion scaled total queue length process, $\hat{Q}_{sf}^{(n)}(t)$, converges weakly to a one-dimensional reflected Brownian motion $\hat{Q}_{sh}$. To be precise, $\hat{Q}_{sf}^{(n)}(t) \xrightarrow{w} \hat{Q}_{sh} = RBM\left(b_1,\ \lambda_1 + \sum_{j\in\mathcal{J}} \mu_j^3\beta_j^2\right)$, as $n \to \infty$.

(ii) For MARO–tree, $\hat{Q}_{st}^{(n)}(t) \xrightarrow{w} \hat{Q}_{sh}$.

(iii) For $M/G/J^h$, $\hat{Q}_{Mh}^{(n)}(t) \xrightarrow{w} \hat{Q}_{sh}$.

Proof.

(i) From (3.20), we have

$$\hat{Y}_f^{(n)}(t) = \nu_1^* \hat{Q}_{sf}^{(n)}(t),$$

where the arrival workload contribution ν_1^* equals $1/\tilde{\mu}$ in this heterogeneous case. Using Theorem 3.3.2, we have

$$\frac{\hat{Q}_{sf}^{(n)}(t)}{\tilde{\mu}} \xrightarrow{w} \text{RBM}\left(\frac{b_1}{\tilde{\mu}}, \frac{\lambda_1 + \sum_{j\in\mathcal{J}} \mu_j^3 \beta_j^2}{\tilde{\mu}^2}\right),$$

which yields the result.

(ii) Using Theorem 3.3.3 and reasoning similar to that in part (i), we have the result.

(iii) A direct application of Theorem 2.2.5 gives the result.

□

3.5 Summary

In this chapter, we have proposed a series of MinDrift affinity routing policies, namely MARO, MARO–$2/k$, MARO–flex and MARO–tree. As resource management strategies, these policies use the solution to a resource allocation LP which aims to maximize the system capacity. As dynamic routing policies, they also use state information to shift workload between servers, trying to minimize the expected increment of the holding cost due to delay in the system. Compared with MinDrift(Q), which requires global state information for every routing decision, the proposed policies significantly reduce the amount of the required state information in making decisions, which benefits from using the solution to the LP. We have shown that the larger the system, the greater the

reduction. It is also noted that the MARO-related policies are robust in the sense that multiplying the mean arrival rates of all task types by the same factor does not change the routing probabilities derived from the LP.

To accommodate different levels of flexibility, three variants of the MARO policy, MARO$-2/k$, MARO$-$flex and MARO$-$tree, are introduced. They reduce further the required state information, either by choosing a smaller number of candidate queues or by allowing fewer flexible arrivals, or both. Using diffusion limits, we have shown that MARO$-$flex and MARO$-$tree have the same heavy traffic optimality properties as does MinDrift(Q) and the optimality is achieved independent of the levels of flexibility. For the special cases of systems with a single task type, MARO$-$flex and MARO$-$tree approach the lower bound of achievable performance by a multi-server single queue.

As a result, we will demonstrate in Chapter 5 that (1) for systems in heavy traffic, a small amount of flexibility can yield significant improvement in system performance; (2) when the system is not operating under heavy traffic, MARO and MARO$-2/k$ achieve better performance than does MinDrift(Q), by making use of an "appropriately" limited amount of state information.

After having compared the resource allocation policies which use full or partial state information, it is natural to ask what can be done without any state information. We will study this logical extreme case in Chapter 4. The results obtained can be useful to compare the performance of the policies that utilize different amounts of state information.

Chapter 4

Resource Allocation with No State Information

In Chapter 3, we proposed several dynamic routing policies, which aim to minimize the delay in an output-queued system by using a *limited* amount of state information. In this chapter, we will propose resource allocation policies for systems where *no* state information is available. The systems under study are equipped with *homogeneous* servers to process multiple types of tasks, while a task's processing time is unknown until its completion. The task inter-arrival times are known to follow exponential distributions, while the task processing times follow general continuous or discrete distributions. The proposed resource allocation policies consist of two parts: one is the routing policy and the other is the pooling strategy which combines several parallel queues into a multi-server single queue, in order to further reduce the delay in the system.

Section 4.1 presents the routing policy which minimizes the delay in a parallel-queue system under heavy traffic, by taking into account the second moments of the task processing times. Section 4.2 compares the pooling strategies that are coupled with the given routing policy. Finally, Section 4.3 discusses extensions of the results to *heterogeneous* server systems with general arrival processes.

4.1 Routing Policy

4.1.1 The Heavy Traffic Model

Consider an output-queued system as defined in Section 3.1, which has J identical servers in parallel to process I types of tasks. Type i tasks arrive according to a Poisson process with rate $\lambda_i > 0$. The processing times of type i tasks at each server are i.i.d. and form the sequence $\{v_{i,m} : m \geq 1\}$. Both $\mu_i = 1/E[v_{i,1}]$ and $\beta_i^2 = Var[v_{i,1}]$ are assumed finite. The local scheduling rule at each server is FCFS. A type i task is routed to one of the queues j immediately upon arrival, with probability $p_{i,j}$. Define the routing probability matrix $P = (p_{i,j})_{I \times J}$. We have

$$\sum_{j=1}^{J} p_{i,j} = 1, \quad \forall i \in \mathcal{I}. \tag{4.1}$$

At the j-th queue, we have a Poisson arrival stream with rate

$$\lambda_{e_j} = \sum_{i=1}^{I} \lambda_i p_{i,j}, \tag{4.2}$$

while the service time distribution of server j has mean

$$\mu_{e_j}^{-1} = \sum_{i=1}^{I} \frac{\lambda_i p_{i,j}}{\lambda_{e_j}} \mu_i^{-1} \tag{4.3}$$

and variance

$$\beta_{e_j}^2 = \sum_{i=1}^{I} \frac{\lambda_i p_{i,j}}{\lambda_{e_j}} (\mu_i^{-2} + \beta_i^2) - \mu_{e_j}^{-2}. \tag{4.4}$$

Let $\rho_i = \lambda_i/\mu_i$ and $\psi_{i,j}^*$ be the average rate at which the j-th server's time is allocated to process type i tasks. Then we have

$$p_{i,j} = \psi_{i,j}^*/\rho_i. \tag{4.5}$$

As has been analyzed in Section 3.4.2, Assumption 3.3.1 holds for this homogeneous server system. So all servers are equally utilized, i.e.,

$$\sum_{i=1}^{I} \psi_{i,j}^* = \rho^* \leq 1, \quad \forall j \in \mathcal{J}, \tag{4.6}$$

where $\rho^* = \sum_{i=1}^{I} \rho_i / J$. Combining (4.5) and (4.6), we have

$$\sum_{i=1}^{I} \rho_i p_{i,j} = \rho^*, \quad \forall j \in \mathcal{J}. \tag{4.7}$$

Define the vector $\lambda = (\lambda_i)_{1\times I}$, the matrices μ and β which consist of J identical column vectors $\vec{\mu} = (\mu_i)_{I\times 1}$ and $\vec{\beta} = (\beta_i^2)_{I\times 1}$, respectively. Given (4.6), we make the following assumption.

Assumption 4.1.1. *The first-order primitives* (λ, μ) *are such that* $\rho^* = 1$.

Given (4.2), (4.3) and (4.7), Assumption 4.1.1 implies $\lambda_{e_j} = \mu_{e_j}$, i.e., each server j is under heavy traffic using the routing matrix P.

To define the heavy traffic regime, now consider a sequence of the systems defined above, indexed by n. For type i tasks in the n-th system, the Poisson arrival rate is $\lambda_i^{(n)}$; the service time distribution has mean μ_i^{-1} and variance β_i^2. We assume that the following conditions hold

$$\lim_{n\to\infty} \lambda_i^{(n)} = \lambda_i, \tag{4.8}$$

and

$$\mathbf{E}\left[v_{i,1}^{2+\epsilon}\right] \equiv d_i < \infty, \tag{4.9}$$

for some $\epsilon > 0$, where $\{v_{i,m} : m \geq 1\}$ is a sequence of i.i.d. random variables formed by the processing times for task type i at each identical server. In addition, the heavy traffic condition

$$\lim_{n\to\infty} \sqrt{n}\left(\lambda_i^{(n)} - \lambda_i\right) = b \tag{4.10}$$

is assumed to be true for all $i \in \mathcal{I}$ for some finite constant b. We define a constant c_j for each $j \in \mathcal{J}$ such that

$$\sum_{i=1}^{I} p_{i,j} b = c_j. \tag{4.11}$$

From (4.2)–(4.4) and (4.8)–(4.9), we have for queue j the following heavy traffic conditions

$$\lim_{n\to\infty} \lambda_{e_j}^{(n)} = \lambda_{e_j}, \tag{4.12}$$

$$\lim_{n\to\infty} \mu_{e_j}^{(n)} = \mu_{e_j}, \tag{4.13}$$

$$\lim_{n\to\infty} \left(\beta_{e_j}^2\right)^{(n)} = \beta_{e_j}^2, \tag{4.14}$$

and

$$\mathbf{E}\left[\left(v_{e_j,1}^{2+\epsilon}\right)^{(n)}\right] = \sum_{i=1}^{I} \frac{p_{i,j}\lambda_i}{\lambda_{e_j}} d_i < \infty, \tag{4.15}$$

for some $\epsilon > 0$, where $\{v_{e_j,m} : m \geq 1\}$ is a sequence of i.i.d. random variables formed by the effective processing times at server j. In addition, from (4.10), (4.11) and Assumption 4.1.1, we have

$$\lim_{n\to\infty} \sqrt{n}\left(\lambda_{e_j}^{(n)} - \mu_{e_j}\right) = c_j. \tag{4.16}$$

Let the queue length process for queue j be given by $Q_j^{(n)}(t)$. We form the diffusion scaled queue length process as

$$\hat{Q}_j^{(n)}(t) = \frac{1}{\sqrt{n}} Q_j^{(n)}(nt). \tag{4.17}$$

From (4.12)-(4.16) and Theorem 2.2.4, we have the following result:

Theorem 4.1.1. *$\hat{Q}_j^{(n)}(t) \xrightarrow{w} \hat{Q}_j = RBM\left(c_j, \sigma_j^2\right)$, as $n \to \infty$, where*

$$\sigma_j^2 = \left(\sum_{i=1}^{I} \lambda_i p_{i,j}\right)^2 \left[\sum_{i=1}^{I} \lambda_i(\mu_i^{-2} + \beta_i^2)p_{i,j}\right]. \tag{4.18}$$

From Theorem 2.1.7, the mean of the stationary distribution of RBM $\hat{Q}_j$ is $\varphi_j = \sigma_j^2/(2|c_j|)$. Since the queue length processes $Q_j^{(n)}(t)$ are mutually independent, we define the weighted total mean

$$\tilde{\varphi} = 2|b| \sum_{j=1}^{J} \varphi_j = \sum_{j=1}^{J} \frac{\left(\sum_{i=1}^{I} \lambda_i p_{i,j}\right)^2 \left(\sum_{i=1}^{I} \theta_i p_{i,j}\right)}{\sum_{i=1}^{I} p_{i,j}}. \tag{4.19}$$

The parameter

$$\theta_i = \lambda_i(\mu_i^{-2} + \beta_i^2) = \lambda_i E[v_i^2] \tag{4.20}$$

includes the second moment of the random variable v_i, from which the processing times of type i tasks are generated.

Our goal is, given the primitives (λ, μ, β) and Assumption 4.1.1, to find an optimal routing matrix P^* which yields the minimal $\tilde{\varphi}$, subject to

conditions (4.7) and (4.1). It can be formulated as a nonlinear programming (NLP) problem

$$\min_{P} \quad \tilde{\varphi}(P) = \sum_{j=1}^{J} \frac{\left(\sum_{i=1}^{I} \lambda_i p_{i,j}\right)^2 \left(\sum_{i=1}^{I} \theta_i p_{i,j}\right)}{\sum_{i=1}^{I} p_{i,j}} \tag{4.21}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \rho_i p_{i,j} = 1, \quad \forall j \in \mathcal{J} \tag{4.22}$$

$$\sum_{j=1}^{J} p_{i,j} = 1, \quad \forall i \in \mathcal{I} \tag{4.23}$$

$$0 \leq p_{i,j} \leq 1, \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}.$$

Before proceeding to solve this NLP, we compare it with the static allocation LP (3.2) given in Section 3.3.1. Since Assumptions 3.3.1 and 4.1.1 hold for homogeneous systems, constraints (4.22) and (4.23) are equivalent to the constraints of LP (3.2), with inequalities (3.3) and (3.4) changed to equalities. Therefore, given the same set of constraints, the routing matrix derived from the solution Ψ^* to LP (3.2) maximizes the system capacity (or throughput), while the solution to NLP (4.21) minimizes the total queue length of the system (or equivalently the delay in the system) in heavy traffic. In some special cases, these two routing matrices are the same [1], but in general, they are different. The difference between the two static routing policies shows that maximizing the throughput does not necessarily result in minimizing the delay in the system. That is why the variance of processing times is taken into account, in order to achieve the latter objective.

[1]For example, given that the processing times are exponentially distributed, $I = J = 2$, $\rho_1 = 0.5$ and $\mu_1/\mu_2 = 2$, the solution to NLP (4.21) is

$$P^* = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}.$$

It is the same as the static routing policy derived from the solution to LP (3.2).

4.1.2 The Resource Allocation NLP

Define

- decision variables $\mathbf{x} = vec(P)$ and augmented variables $\mathbf{y} = [y_1, ..., y_{3J}]^T$;
- parameters $\rho = [\rho_1, ..., \rho_I]$ and $\theta = [\theta_1, ..., \theta_I]$, where $\rho_i = \lambda_i/\mu_i \geq 0$ and as defined in (4.20) $\theta_i \geq 0$. (By convention, $1/\mu_i = 0$ if $\mu_i = 0$.)

NLP (4.21) is reformulated as

$$\min_{(\mathbf{x},\mathbf{y})} \quad \tilde{\varphi}(\mathbf{y}) = \sum_{j=1}^{J} \left(y_j^2\right)\left(y_{j+J}\right)\left(y_{j+2J}^{-1}\right) \tag{4.24}$$

$$\text{s.t.} \quad y_j - \sum_{i=1}^{I} \lambda_i x_{(j-1)I+i} = 0, \quad \forall j \in \mathcal{J} \tag{4.25}$$

$$y_{j+J} - \sum_{i=1}^{I} \theta_i x_{(j-1)I+i} = 0, \quad \forall j \in \mathcal{J} \tag{4.26}$$

$$y_{j+2J} - \sum_{i=1}^{I} x_{(j-1)I+i} = 0, \quad \forall j \in \mathcal{J} \tag{4.27}$$

$$\sum_{i=1}^{I} \rho_i x_{(j-1)I+i} = 1, \quad \forall j \in \mathcal{J} \tag{4.28}$$

$$\sum_{j=1}^{J} x_{(j-1)I+i} = 1, \quad \forall i \in \mathcal{I} \tag{4.29}$$

$$0 \leq x_n \leq 1, \quad \forall n \in \{1, ..., IJ\} \tag{4.30}$$

$$0 \leq y_j \leq \sum_{i=1}^{I} \lambda_i, \quad 0 \leq y_{j+J} \leq \sum_{i=1}^{I} \theta_i,$$
$$0 \leq y_{j+2J} \leq I, \quad \forall j \in \mathcal{J}. \tag{4.31}$$

Constraints (4.25)–(4.27) are merely substitutions of variables. Constraints (4.28) and (4.29) are equivalent to constraints (4.22) and (4.23), respectively. All the variables are bounded by (4.30) and (4.31). Consequently, NLP (4.24) consists of linear equality constraints and a non-convex objective function.

Note that the objective function is a posynomial [2] and the constraint functions can also be transformed to posynomials. This suggests that NLP (4.24) might be transformed into a geometric programming (GP) problem, which in turn can be solved using convex optimization techniques [9]. However, our attempt to solve NLP (4.24) with the GP solver in the MOSEK optimization toolbox [53] was not successful (see Appendix A).

Based on the benchmarking results provided by Dolan et al. in [20] and Morales et al. in [52], we choose a general purpose NLP solver, KNITRO [18], to solve the resource allocation problem (4.24). Unlike some other NLP solvers, e.g., MINOS [54] or SNOPT [27], which are designed to find a locally optimal solution which is close to the starting point of a feasible solution $\mathbf{x}$, KNITRO features a multi-start procedure [37] so that the solution returned is the local optimum with the best objective function value. This improves the probability of finding the global optimum for NLP.

Here we give an example of computing the optimal static routing policy for a homogeneous system in heavy traffic. This example is abstracted from the Grid application in Section 5.2.1.

Example 4.1.1. *Let $I = 5$ and $J = 6$. The first-order primitives are*

$$\lambda = \begin{bmatrix} 39.84 & 13.47 & 15.15 & 0.94 & 2.06 \end{bmatrix}$$

and

$$\vec{\mu} = \begin{bmatrix} 16.7 & 30.4 & 18.9 & 3.0 & 1.0 \end{bmatrix}^T,$$

so that Assumption 4.1.1 holds.

Assume that the processing time distributions are exponential, i.e., the parameter θ_i defined in (4.20) is equal to $2\lambda_i/\mu_i^2$ for task type i. From the analysis in Section 3.4.2, we know that a natural starting point is the routing matrix P^0 with all of the elements being $1/J$. Using KNITRO, the global

[2] Let $\mathbf{x} = [x_1, ..., x_n]$ be a vector with components x_i, $i = 1, ..., n$. A function of the form

$$f(\mathbf{x}) = \sum_{k=1}^{K} c_k \prod_{i=1}^{n} x_i^{a_{i,k}},$$

where $c_k > 0$ and $a_{i,k} \in \mathbb{R}$, is called a posynomial (function). If $K = 1$, $f(\mathbf{x})$ is a monomial.

optimal solution to NLP (4.24) *is obtained to be*

$$P^* = \begin{bmatrix} & 0.20 & & 0.24 & 0.29 & 0.27 \\ & & & 0.30 & 0.70 & \\ & 0.64 & & 0.36 & & \\ & & & & & 1.00 \\ 0.49 & & 0.49 & & & 0.02 \end{bmatrix}, \tag{4.32}$$

which yields the objective function value

$$\tilde{\varphi}^*_{exp} = 178.6. \tag{4.33}$$

Several observations can be made from this example.

Firstly, if the processing time distributions are changed such that the squared coefficient of variation C_s^2 is altered by a factor of k for all task types, the optimal static routing policy solution is the same as that given in (4.32), while the objective function value $\tilde{\varphi}^*$ is differentiated by a factor of $(k+1)/2$ from that of the exponential processing time case. This can be seen by rewriting the parameter θ_i in (4.19) as

$$\theta_i = \lambda_i(1 + C_{s,i}^2)/\mu_i^2.$$

For example, let the processing time distributions be Erlang$-k$ for all task types, then we have $C_{s,i}^2 = 1/k$ for all $i \in \mathcal{I}$. Suppose that the optimal solution P^* yields $\tilde{\varphi}^*_{exp}$ for exponential processing times. Then the same matrix P^* is also the optimal solution for the Erlang$-k$ processing times and the objective function value is

$$\tilde{\varphi}^*_{E_k} = \left(\frac{1+1/k}{2}\right)\tilde{\varphi}^*_{exp}.$$

Secondly, the routing matrix P^0 derived from LP (3.2) is used as the starting point to solve NLP (4.24), however, the resulting optimal routing matrix P^* is quite different. This implies that maximizing the system throughput does not necessarily lead to minimizing the delay in the system. What is the same between P^0 and P^* is that column permutations of each routing probability matrix keep the corresponding objective function value optimal, since the servers are identical.

Thirdly, since NLP (4.21) and LP (3.2) have the same set of linear constraints, we have the following proposition to quantify the number of positive elements N_p in a solution P.

Proposition 4.1.1. *There exists a feasible solution P, whose associated N_p satisfies* $\max(I, J) \leq N_p \leq I + J$.

Proof. The proof is similar to that of Proposition 3.3.1. The first inequality follows from the fact that otherwise there exists either at least one task type that is not processed by any servers or at least one idle server. If there were an idle server, the objective function value $\tilde{\varphi}^*$, which corresponds to the total scaled mean queue length of the system in heavy traffic, could not have been a minimum.

The second inequality results from the generation of the basic feasible solution of NLP (4.21). Since the NLP has only $I+J$ linear equality constraints and always has a non-zero feasible solution which contains IJ variables, then a basic feasible solution (BFS) exists. The BFS contains $I + J$ basic variables which correspond to the linearly independent constraints and $IJ - (I + J)$ nonbasic variables which are zero. Then the matrix P has at least $IJ-(I+J)$ elements equal to zero. Equivalently, the number of positive elements is at most $I + J$. □

Proposition 4.1.1 implies that the matrix P contains many zero elements (especially when the matrix size is large). Although the feasible solution which contains the maximum number of zeros might not be the optimum, it is found that in many cases, the number of zeros in the optimal solution is close to the maximum. For example, the number of positive elements in the optimal solution P^* in (4.32) is 12, close to $I + J = 10$. This implies that using the optimal static routing policy, most servers need only be capable of processing a small subset of the task types. This is desirable when it is expensive to maintain highly flexible servers.

Finally, if more than one column of P^* are the same, e.g., columns 1 and 3 in (4.32), the corresponding servers can be pooled into one queue to further reduce the total mean queue length of the system. To be specific, given $P^* = (p^*_{i,j})_{I\times J}$, suppose there exists a subset of queues $\mathcal{J}_k$ satisfying the following condition:

Condition 4.1.1. *For any two queues j, $j' \in \mathcal{J}_k \subseteq \mathcal{J}$, $p^*_{i,j} = p^*_{i,j'}$ is true for all $i \in \mathcal{I}$.*

We can construct a single queue j_k by pooling these $|\mathcal{J}_k|$ parallel queues together, with $|\mathcal{J}_k|$ arrival streams, each being a Poisson process with rate λ_{e_j} as defined in (4.2). Consider a sequence of such $|\mathcal{J}_k|$-server single queues indexed by n. For the n-th system, let the queue length process of queue j_k be $Q_{j_k}^{(n)}(t)$. Its diffusion scaled version $\hat{Q}_{j_k}^{(n)}(t)$ is formed in the same fashion as (4.17). The following theorem says that the scaled queue length process for the pooled queue converges to the same reflected Brownian motion as that for any one of the queues to be pooled, except that the RBM drift and variance are increased by a factor of $|\mathcal{J}_k|$.

Theorem 4.1.2. *If for any queue $j \in \mathcal{J}_k$, $\hat{Q}_j^{(n)}(t) \xrightarrow{w} RBM\big(c_j,\ \sigma_j^2\big)$, as $n \to \infty$, then for the multi-server queue j_k, $\hat{Q}_{j_k}^{(n)}(t) \xrightarrow{w} RBM\big(|\mathcal{J}_k|c_j,\ |\mathcal{J}_k|\sigma_j^2\big)$, as $n \to \infty$.*

Proof. The proof follows directly from Theorems 2.2.5 and 4.1.1. □

Let the mean of the stationary distribution of the corresponding RBM be φ_j and φ_{j_k}, respectively. We have $\varphi_{j_k} = \sum_{j \in \mathcal{J}_k} \varphi_j / |\mathcal{J}_k|$. This implies that by pooling the $|\mathcal{J}_k|$ parallel queues, the performance of the subsystem can be improved by a factor of $|\mathcal{J}_k|$ in heavy traffic. For example, using the matrix P^* in (4.32), we can pool servers 1 and 3 together in a single queue. The objective function value is reduced to 174.4.

In Appendix B, we propose a heuristic to obtain a suboptimal routing matrix (denoted as P^H) that satisfies the constraints (4.28) and (4.29) in NLP (4.21). When the system size $I \times J$ is large, the matrix P^H has more columns that satisfy Condition 4.1.1 than the optimal solution P^* to NLP (4.21). Therefore, if an NLP solver is not available, P^H is useful to try to increase the degree of pooling.

4.2 Pooling Strategies

There are three kinds of pooling: no pooling, full pooling and partial pooling. The J parallel queue system discussed in Section 4.1.1 involves no pooling, where only the static routing policy is used. Full pooling is to pool all the

queues together into one queue, so that a J-server single queue is formed and no routing is needed. Given the optimal routing policy derived from NLP (4.21), partial pooling is to pool each subset of the J queues, which satisfies Condition 4.1.1, into a multi-server single queue, in order to minimize the total mean queue length of the system. Given the primitives (λ, μ, β), a pooling strategy is to decide which kind of pooling should be applied, so that the total mean queue length of the system is minimized in heavy traffic.

4.2.1 Full Pooling

Using the system described in Section 4.1.1, a multi-server-single-queue system is constructed by pooling the parallel queues into one queue. The local scheduling rule at the J identical servers is FCFS. The arrivals of the I types of tasks follow a Poisson process with rate

$$\tilde{\lambda} = \sum_{i=1}^{I} \lambda_i. \tag{4.34}$$

The service time distributions of the J servers are identical with mean

$$\mu_e^{-1} = \sum_{i=1}^{I} \frac{\lambda_i}{\tilde{\lambda}} \mu_i^{-1} \tag{4.35}$$

and variance

$$\beta_e^2 = \sum_{i=1}^{I} \frac{\lambda_i}{\tilde{\lambda}} (\mu_i^{-2} + \beta_i^2) - \mu_e^{-2}. \tag{4.36}$$

For the first-order primitives (λ, μ), Assumption 4.1.1 implies

$$\sum_{i=1}^{I} \frac{\lambda_i}{\mu_i} = J.$$

So we have $\tilde{\lambda} = J\mu_e$, i.e., this multi-server single queue is under heavy traffic.

In the heavy traffic regime, we assume that (4.8)–(4.10) hold. Then from (4.34)–(4.36), we have

$$\lim_{n\to\infty} \tilde{\lambda}^{(n)} = \tilde{\lambda}, \tag{4.37}$$

$$\lim_{n\to\infty} \mu_e^{(n)} = \mu_e, \tag{4.38}$$

$$\lim_{n\to\infty} \left(\beta_e^2\right)^{(n)} = \beta_e^2, \tag{4.39}$$

and

$$\mathbf{E}\left[\left(v_{e,1}^{2+\epsilon}\right)^{(n)}\right] = \sum_{i=1}^{I} \frac{\lambda_i\, d_i}{\tilde{\lambda}} < \infty, \tag{4.40}$$

for some $\epsilon > 0$, where $\{v_{e,m} : m \geq 1\}$ is a sequence of i.i.d. random variables formed by the effective processing times at each server. In addition, we have

$$\lim_{n\to\infty} \sqrt{n}\left(\tilde{\lambda}^{(n)} - J\mu_e\right) = c_f, \tag{4.41}$$

where $c_f = bI$ is a constant.

Let $\hat{Q}_f^{(n)}(t)$ be the diffusion scaled queue length process. From (4.37)–(4.41) and Theorem 2.2.5, we have the following result:

Theorem 4.2.1. *$\hat{Q}_f^{(n)}(t) \xrightarrow{w} \hat{Q}_f = RBM\left(c_f,\ \sigma_f^2\right)$, as $n \to \infty$, where*

$$\sigma_f^2 = \frac{1}{J^2}\left(\sum_{i=1}^{I} \lambda_i\right)^2 \left[\sum_{i=1}^{I} \lambda_i(\mu_i^{-2} + \beta_i^2)\right]. \tag{4.42}$$

From Theorem 2.1.7, the mean of the stationary distribution of the RBM $\hat{Q}_f$ is $\varphi_f = \sigma_f^2/(2|c_f|)$. We define the weighted mean

$$\tilde{\varphi}_f = 2|b|\,\varphi_f = \frac{1}{IJ^2}\left(\sum_{i=1}^{I} \lambda_i\right)^2 \left(\sum_{i=1}^{I} \theta_i\right), \tag{4.43}$$

where $\theta_i = \lambda_i E[v_i^2]$ is the same as (4.20).

To compare full pooling with no pooling, we denote the weighted total mean defined in (4.19) as $\tilde{\varphi}_n$ (which stands for *no* pooling). Comparing (4.43) with (4.19), it can be seen that if the optimal routing matrix P_n^* for no pooling has all the elements being $1/J$, Theorem 4.2.1 is a special case of Theorem 4.1.2 with $\mathcal{J}_k = \mathcal{J}$. Then we have $\tilde{\varphi}_n^* = J\tilde{\varphi}_f$, i.e., full pooling is better than no pooling.

However, there are cases where the performance of no pooling is superior than that of full pooling, when P_n^* has other structures. Table 4.1 shows two such cases, where Assumption 4.1.1 holds for the first-order primitives (λ, μ), i.e., $\sum_{i=1}^{I} \rho_i = J = 3$. Since $\tilde{\varphi}_n^* < \tilde{\varphi}_f$, the no pooling structure outperforms the full pooling structure in heavy traffic.

Table 4.1: Cases of no pooling superior than full pooling, $I = J = 3$, exponential processing times

Cases	i	ρ_i	μ_i	λ_i	$P_n^* = (p_{i,j})_{I\times J}$			$\tilde{\varphi}_n^*$	$\tilde{\varphi}_f$
1	1	1.10	20.00	22.00	0.07	0.91	0.02		
	2	0.10	2.00	0.20	1.00				
	3	1.80	0.20	0.36	0.46	0.54		71.2	343.3
2	1	1.80	20.00	36.00	0.44		0.56		
	2	0.10	2.00	0.20	1.00				
	3	1.10	0.20	0.22	0.09	0.91		274.6	554.2

Table 4.2: Cases of full pooling superior than no pooling, $I = J = 3$, exponential processing times

Cases	i	ρ_i	μ_i	λ_i	$P_n^* = (p_{i,j})_{I\times J}$			$\tilde{\varphi}_n^*$	$\tilde{\varphi}_f$
1	1	0.10	20.00	2.00	1.00				
	2	1.80	2.00	3.60	0.50		0.50		
	3	1.10	0.20	0.22		0.91	0.09	19.8	16.1
2	1	1.80	20.00	36.00	0.56	0.44			
	2	1.10	2.00	2.20		0.18	0.82		
	3	0.10	0.20	0.02			1.00	195.7	123.4

4.2.2 Partial Pooling

Using the same matrix μ and the task type loads ρ_i in Table 4.1, but with different combinations (which result in different vectors λ), Table 4.2 shows two cases where the no pooling structure is outperformed by the full pooling structure ($\tilde{\varphi}_n^* > \tilde{\varphi}_f$). Now the question is given the optimal routing matrix P_n^* which yields the no pooling structure outperformed by the full pooling structure, is there a partial pooling structure that outperforms the full pooling structure?

The answer is depending on the primitives (λ, μ, β), pooling each subset of the J queues that satisfies Condition 4.1.1, into a multi-server single queue, can yield a partial pooling structure that outperforms the full pooling structure.

Define the set $\mathcal{K} = \{1, ..., K\}$. Suppose that there exists a partition of $\mathcal{J}$ where $\mathcal{J} = \mathcal{J}_0 \cup \left(\bigcup_{k=1}^{K} \mathcal{J}_k\right)$ and $\mathcal{J}_k \cap \mathcal{J}_{k'} = \emptyset, \forall k, k' \in \mathcal{K} \cup \{0\}$. Each server in the subset $\mathcal{J}_0$ maintains its own queue, while parallel queues within each subset $\mathcal{J}_k$ $(k \neq 0)$ satisfy Condition 4.1.1 and are pooled into a multi-server queue j_k. Therefore, the probability with which type i tasks are routed to queue j_k is $p_{i,j_k} = |\mathcal{J}_k| p_{i,k}$, where $p_{i,k} = p_{i,j}^*$, for any $j \in \mathcal{J}_k$.

From Theorem 4.1.2, the mean of the stationary distribution of the RBM $\hat{Q}_{j_k}$ is equal to that of the RBM $\hat{Q}_j$, for any queue $j \in \mathcal{J}_k$. Since the RBMs $\hat{Q}_{j_k}$ $(k \in \mathcal{K})$ are mutually independent, we define the weighted total mean

$$\begin{aligned} \tilde{\varphi}_p &= 2|b| \left(\sum_{j \in \mathcal{J}_0} \varphi_j + \sum_{k \in \mathcal{K}} \varphi_{j_k} \right) \\ &= \sum_{j \in \mathcal{J}_0} \frac{\left(\sum_{i \in \mathcal{I}} \lambda_i p_{i,j}^*\right)^2 \left(\sum_{i \in \mathcal{I}} \theta_i p_{i,j}^*\right)}{\sum_{i \in \mathcal{I}} p_{i,j}^*} + \\ &\quad \sum_{k \in \mathcal{K}} \frac{\left(\sum_{i \in \mathcal{I}} \lambda_i p_{i,k}\right)^2 \left(\sum_{i \in \mathcal{I}} \theta_i p_{i,k}\right)}{\sum_{i \in \mathcal{I}} p_{i,k}}. \end{aligned} \tag{4.44}$$

If $\mathcal{J}_k = \emptyset$ for all $k \neq 0$, we have $\mathcal{J}_0 = \mathcal{J}$ and $\tilde{\varphi}_p = \tilde{\varphi}_n^*$. Otherwise, Theorem 4.1.2 implies $\tilde{\varphi}_p < \tilde{\varphi}_n^*$, which means the total mean queue length of the system can be reduced further in heavy traffic with partial pooling.

Table 4.3 shows an example, where the routing matrix P_n^H for no pooling yields performance worse than that of full pooling ($\tilde{\varphi}_n^H > \tilde{\varphi}_f$). However, given P_n^H, by pooling two subsets of the queues: queues 1 and 2, queues 5 and 6, respectively, partial pooling is better than full pooling ($\tilde{\varphi}_f > \tilde{\varphi}_p$).

Table 4.3: A case of partial pooling superior than full pooling, $I = J = 6$, exponential processing times

i	ρ_i	μ_i	λ_i	$P_n^H = (p_{i,j})_{I\times J}$						$\tilde{\varphi}_n^H$	$\tilde{\varphi}_f$	$\tilde{\varphi}_p$
1	0.40	2.01	0.80				0.33	0.33	0.33			
2	0.88	2.05	1.80					0.50	0.50			
3	2.90	2.02	5.86	0.34	0.34		0.10	0.11	0.11			
4	0.60	2.00	1.20				1.00					
5	0.22	1.95	0.43					0.50	0.50			
6	1.00	78.00	78.00			1.00				187.6	178.9	170.3

Therefore, given the primitives (λ, μ, β), to choose a pooling strategy which minimizes the total mean queue length in heavy traffic, we have the following procedure:

1. Solve NLP (4.21) and obtain the optimal objective function value $\tilde{\varphi}_n^*$ and the routing matrix P_n^*.

2. Given P_n^*, pool each subset $\mathcal{J}_p$ of the J queues that satisfies Condition 4.1.1, into a multi-server single queue and calculate the weighted total mean for partial pooling $\tilde{\varphi}_p$ in (4.44). If none of the subsets satisfies Condition 4.1.1, we have $\tilde{\varphi}_p = \tilde{\varphi}_n^*$, otherwise we have $\tilde{\varphi}_p < \tilde{\varphi}_n^*$.

3. Calculate the weighted mean for full pooling $\tilde{\varphi}_f$ using (4.43).

4. If $\tilde{\varphi}_p > \tilde{\varphi}_f$, the full pooling structure should be applied. Otherwise, the partial pooling structure should be applied. In the latter case, if $\tilde{\varphi}_p = \tilde{\varphi}_n^*$, no pooling is needed.

When choosing between full pooling and partial pooling, there is again a tradeoff between the system performance and the cost of maintaining highly flexible servers. Full pooling requires each server capable of processing all types of tasks, while partial pooling requires less flexibility.

Finally, we present in Figure 4.1 an algorithm to find the partition of the J parallel queues according to an equivalence condition which is exactly as or "close to" Condition 4.1.1, given the optimal solution P^* of NLP (4.21). In the worst case, the procedure `partition`(P) makes $O(J^2)$ comparisons between the J columns of the matrix P. In the procedure `Check_Arrays`$(p,\ q)$, if ϵ is not larger than the machine precision, then queues p and q exactly satisfy Condition 4.1.1. To increase the degree of pooling and obtain a potential smaller total mean $\tilde{\varphi}_p$, we may want to relax Condition 4.1.1 by choosing a larger ϵ (e.g., 10^{-2}). Then the partition becomes less fine, i.e., the number of subsets decreases. Consequently, more servers are required to be capable of processing more types of tasks.

4.3 Extensions

In this section, we discuss the possible extensions of the results obtained for homogeneous server systems to heterogeneous systems. Section 4.3.1 shows that for heterogeneous systems with Poisson arrival processes, where the system load does not approach one and the Pollaczek-Kintchine formula can be applied, the NLP used to obtain the optimal routing matrix is no more complicated than that in the heavy traffic case. In Section 4.3.2, the results are extended to heterogeneous systems with generic arrival processes in the heavy traffic regime.

4.3.1 Moderate Traffic

Consider J heterogeneous servers in parallel to process I types of tasks. The arrivals of task type i follow a Poisson process with rate $\lambda_i > 0$. The processing times of type i tasks at server j are i.i.d. and form the sequence $\{v_{i,j,m} : m \geq 1\}$. Both $\mu_{i,j} = 1/E[v_{i,j,1}]$ and $\beta_{i,j}^2 = Var[v_{i,j,1}]$ are assumed finite. The local

```
1   proc partition(P)
2   {   y, z: column (or queue) indices of the matrix P with size I × J;
3       k: index of the subset S[k] ⊆ {1, ..., J};
4       j: queue index of the first element in the set S[k];
5       part[J]: array of length J, element part[y] = 1 if queue y is in some set S[k];
6
7       k = 0;   j = 1;   part[1, ..., J] = 0;   // initialization
8       while ( Sum(part) < J )
9       {   k = k + 1;   S[k] = {j};   z = j + 1;
10          while (z ≤ J)
11          {   if (part[z] == 0)
12              {   Equal = 0;
13                  Equal = Check_Arrays(P[:, j], P[:, z]);
14                  if (Equal == 1)   // (P[i, j] − P[i, z]) < ε, ∀i ∈ {1, ..., I}.
15                  {   S[k] = S[k] ∪ {z};
16                      part[z] = 1;
17                  }
18              }
19              z = z + 1;
20          }
21          y = j + 1;
22          while (y ≤ J AND part[y] == 1)   y = y + 1;
23          j = y;   // find the first element of the new set S[k];
24      }
25      return(S[1], ..., S[k]);
26  }
27  // proc Sum(p) returns the sum of array p's elements.
28  // proc Check_Arrays(p, q) returns 1 if (p[i] − q[i]) < ε, ∀i ≤ length(p); 0, otherwise.
```

Figure 4.1: Procedure for partitioning the routing matrix P

scheduling rule at each server is FCFS. A type i task is routed to queue j immediately upon arrival, with probability $p_{i,j}$. So we have

$$\sum_{j=1}^{J} p_{i,j} = 1, \quad \forall i \in \mathcal{I}. \tag{4.45}$$

At the j-th queue, we have a Poisson arrival stream with rate

$$\lambda_{e_j} = \sum_{i=1}^{I} \lambda_i p_{i,j},$$

while the effective service time distribution of server j has mean

$$\mu_{e_j}^{-1} = \sum_{i=1}^{I} \frac{\lambda_i p_{i,j}}{\lambda_{e_j}} \mu_{i,j}^{-1}$$

and variance

$$\beta_{e_j}^2 = \sum_{i=1}^{I} \frac{\lambda_i p_{i,j}}{\lambda_{e_j}} (\mu_{i,j}^{-2} + \beta_{i,j}^2) - \mu_{e_j}^{-2}.$$

Let $\rho_{i,j} = \lambda_i/\mu_{i,j}$ and $\rho_{e_j} = \lambda_{e_j}/\mu_{e_j}$. We have the effective offered load for server j to be

$$\rho_{e_j} = \sum_{i=1}^{I} \rho_{i,j} p_{i,j}. \tag{4.46}$$

In general, we have $\rho_{e_j} \neq \rho_{e_{j'}}$ for any two queues $j \neq j'$ in moderate traffic. This is different from the heavy traffic case (cf. (4.7)).

Since $0 < \rho_{e_j} < 1$, for all $j \in \mathcal{J}$, by applying the Pollaczek-Kintchine formula, we have the mean queue length of queue j to be

$$\begin{aligned} Q_j &= \rho_{e_j} + \frac{\rho_{e_j}^2 + \lambda_{e_j}^2 \beta_{e_j}^2}{2(1-\rho_{e_j})} \\ &= \sum_{i=1}^{I} \rho_{i,j} p_{i,j} + \frac{\left(\sum_{i=1}^{I} \lambda_i p_{i,j}\right)\left(\sum_{i=1}^{I} \theta_{i,j} p_{i,j}\right)}{2(1-\sum_{i=1}^{I} \rho_{i,j} p_{i,j})}, \end{aligned} \tag{4.47}$$

where $\theta_{i,j} = \lambda_i(\mu_{i,j}^{-2} + \beta_{i,j}^2) = \lambda_i E[v_{i,j}^2]$.

Let $\tilde{Q} = \sum_{j=1}^{J} Q_j$ denote the total mean queue length of the system. To find an optimal routing matrix P^* which yields the minimal $\tilde{Q}$, subject to condition (4.45), we can formulate an NLP as follows. Define

- decision variables $\mathbf{x} = vec(P)$ and augmented variables $\mathbf{y} = [y_1, ..., y_{3J}]^T$;
- parameters $\lambda = (\lambda_i)_{1\times I}$, $\rho = (\rho_{i,j})_{I\times J}$ and $\theta = (\theta_{i,j})_{I\times J}$, where $\lambda_i > 0$, for all $1 \le i \le I$ and $\rho_{i,j} \ge 0$ and $\theta_{i,j} \ge 0$, for all $1 \le i \le I$, $1 \le j \le J$. (By convention, $\rho_{i,j} = 0$ and $\theta_{i,j} = 0$ if $\mu_{i,j} = 0$.)

The NLP is

$$\min_{(\mathbf{x},\mathbf{y})} \quad \tilde{Q}(\mathbf{y}) = \frac{1}{2}\sum_{j=1}^{J}(y_j)\,(y_{j+J})\,(y_{j+2J}^{-1}) + \sum_{j=1}^{J}(1 - y_{j+2J}) \tag{4.48}$$

$$\text{s.t.} \quad y_j - \sum_{i=1}^{I}\lambda_i x_{(j-1)I+i} = 0, \quad \forall j \in \mathcal{J}$$

$$y_{j+J} - \sum_{i=1}^{I}\theta_{i,j} x_{(j-1)I+i} = 0, \quad \forall j \in \mathcal{J}$$

$$y_{j+2J} + \sum_{i=1}^{I}\rho_{i,j} x_{(j-1)I+i} = 1, \quad \forall j \in \mathcal{J}$$

$$\sum_{j=1}^{J} x_{(j-1)I+i} = 1, \quad \forall i \in \mathcal{I}$$

$$0 \le x_n \le 1, \quad \forall n \in \{1, ..., IJ\}$$

$$0 \le y_j \le \sum_{i=1}^{I}\lambda_i, \quad 0 \le y_{j+J} \le \sum_{i=1}^{I}\theta_{i,j},$$

$$\max\left(1 - \sum_{i=1}^{I}\rho_{i,j},\, 0\right) \le y_{j+2J} \le 1, \quad \forall j \in \mathcal{J}.$$

Comparing NLP (4.48) with NLP (4.24), NLP (4.48) has J fewer linear equality constraints and its objective function remains non-convex, with the highest order power being one degree lower.

Although the results of random routing can be extended to systems under moderate traffic, it is not easy to obtain the corresponding pooling strategies. To estimate the mean queue length of multi-server queues which have generic processing time distributions, approximation techniques with take into account the traffic load factor are more complicated than heavy traffic analysis (for example, using Allen-Cunneen approximation requires computing the Erlang-C formula).

Next, we give an example of computing the optimal random routing policy for heterogeneous systems in moderate traffic. The example is abstracted from the Grid application in Section 5.2.1.

Example 4.3.1. *Let $I = 5$ and $J = 6$. The first-order primitives are*

$$\lambda_{M_1} = \begin{bmatrix} 48.75 & 16.48 & 18.54 & 1.14 & 2.52 \end{bmatrix}$$

and

$$\mu = \begin{bmatrix} 16.7 & 24.8 & 24.2 & 29.0 & 25.6 & 48.3 \\ 30.4 & 48.3 & 77.7 & 83.6 & 135.9 & 144.9 \\ 18.9 & 24.2 & 48.3 & 45.8 & 72.5 & 72.5 \\ 3.0 & 3.0 & 7.6 & 7.6 & 8.3 & 8.7 \\ 1.0 & 1.1 & 3.0 & 2.9 & 3.0 & 3.0 \end{bmatrix},$$

so that the solution to LP (3.2) *is $\rho^* = 0.5$. Let the starting point be*

$$P^0 = \begin{bmatrix} 0.17 & 0.25 & & 0.08 & & 0.50 \\ & & & & 1.00 & \\ & & & & 1.00 & \\ & & & 0.11 & 0.89 & \\ & & 0.60 & 0.40 & & \end{bmatrix},$$

which is derived from the solution to LP (3.2) *using* (4.56). *Assume that the processing times are exponentially distributed. Then the global optimal solution to NLP* (4.48) *is obtained to be*

$$P^*_{M_1} = \begin{bmatrix} 0.13 & 0.25 & & & & 0.62 \\ & & & & 1.00 & \\ & & & & 1.00 & \\ & & & 1.00 & & \\ & & 0.62 & 0.38 & & \end{bmatrix}. \tag{4.49}$$

The effective offered loads for the servers are given by the vector

$$\rho^*_{e,M_1} = (\rho^*_{e_j})_{1\times J} = \begin{bmatrix} 0.37 & 0.48 & 0.52 & 0.48 & 0.38 & 0.63 \end{bmatrix}.$$

To see the impact of system load, we increase the mean arrival rates to be $\lambda_{M_2} = 1.9 \times \lambda_{M_1}$ and keep the processing times unchanged, so that the solution to LP (3.2) *is $\rho^* = 0.95$. Then using the same starting point P^0, the*

global optimal solution to NLP (4.48) *is obtained to be*

$$P^*_{M_2} = \begin{bmatrix} 0.17 & 0.25 & & & 0.06 & 0.52 \\ & & & & 1.00 & \\ & & & & 1.00 & \\ & & & 1.00 & & \\ & & 0.60 & 0.40 & & \end{bmatrix}. \tag{4.50}$$

The effective offered loads are given by the vector

$$\rho^*_{e,M_2} = \begin{bmatrix} 0.95 & 0.96 & 0.96 & 0.95 & 0.95 & 0.96 \end{bmatrix}.$$

It is known from (4.47) that the optimal solutions (4.49) and (4.50) do not change if the processing time distributions are changed such that the squared coefficient of variation is increased by a factor of $k > 1$, for all task types at each server. Similar to the observation made for the heavy traffic case, the optimal random routing policy which minimizes the delay in the system by using the first and second moments of the processing times is different from the static routing policy which maximizes the system throughput without using the second moments of the processing times. When the system load approaches one, the effective offered loads of the heterogeneous servers become closer to each other and the optimal routing matrix also changes correspondingly.

4.3.2 General Arrivals

Consider an output-queued system with processing time distributions and local scheduling rule as described in Section 4.3.1, but with different arrival processes. The inter-arrival times of type i tasks are i.i.d. and form the sequence $\{u_{i,m} : m \geq 1\}$. Both $\lambda_i = 1/E[u_{i,1}]$ and $\alpha_i^2 = Var[u_{i,1}]$ are assumed finite. A type i task is routed to queue j immediately upon arrival, with probability $p_{i,j}$.

Let the random variable $\tilde{u}_{i,j} = \sum_{k=1}^{N_{i,j}} u_{i,k}$ be the generic inter-arrival times of type i tasks at server j, where the random variable $N_{i,j}$ is geometrically distributed with parameter $p_{i,j}$. We have

$$E[\tilde{u}_{i,j}] = \lambda_{i,j}^{-1} = (\lambda_i p_{i,j})^{-1}, \tag{4.51}$$

$$Var[\tilde{u}_{i,j}] = \alpha_{i,j}^2 = \frac{\alpha_i^2}{p_{i,j}} + \frac{1}{\lambda_i^2} \cdot \frac{1 - p_{i,j}}{p_{i,j}^2}. \tag{4.52}$$

Let $\lambda_{e_j} = \sum_{i=1}^{I} \lambda_i p_{i,j}$. The effective service times at server j have mean

$$\mu_{e_j}^{-1} = \sum_{i=1}^{I} \frac{\lambda_{i,j}}{\lambda_{e_j}} \mu_{i,j}^{-1} \tag{4.53}$$

and variance

$$\beta_{e_j}^2 = \sum_{i=1}^{I} \frac{\lambda_{i,j}}{\lambda_{e_j}} (\mu_{i,j}^{-2} + \beta_{i,j}^2) - \mu_{e_j}^{-2}. \tag{4.54}$$

Define the vector $\lambda = (\lambda_i)_{1\times I}$ and the matrices $\mu = (\mu_{i,j})_{I\times J}$, $\beta = (\beta_{i,j})_{I\times J}$. Assume that Assumption 3.3.1 holds for the first-order primitives (λ, μ), i.e.,

$$\begin{cases} \sum_{j=1}^{J} \psi_{i,j}^* \mu_{i,j} = \lambda_i, & \forall i \in \mathcal{I}, \\ \sum_{i=1}^{I} \psi_{i,j}^* = \rho^*, & \forall j \in \mathcal{J}, \end{cases} \tag{4.55}$$

where $\psi_{i,j}^*$ is the average rate at which the j-th server's time is allocated to process type i tasks and ρ^* is the long-run utilization of each server j, which approaches one in this case.

Since the processing capacity for each task type equals its arrival rate, we have

$$p_{i,j} = \psi_{i,j}^* \mu_{i,j} / \lambda_i. \tag{4.56}$$

Let $\rho_{i,j} = \lambda_i / \mu_{i,j}$. Combining (4.55) and (4.56), we have

$$\sum_{i=1}^{I} \rho_{i,j} p_{i,j} = 1, \quad \forall j \in \mathcal{J}. \tag{4.57}$$

This implies that using the routing matrix P, each single-server queue j with I arrival streams is under heavy traffic, i.e., $\lambda_{e_j} = \mu_{e_j}$.

Now consider a sequence of systems as defined above, indexed by n. For type i tasks in the n-th system, the inter-arrival time distribution has mean $(\lambda_i^{-1})^{(n)}$ and variance $(\alpha_i^2)^{(n)}$; the service time distribution has mean $\mu_{i,j}^{-1}$ and variance $\beta_{i,j}^2$. For each task type i, we assume that the following conditions hold

$$\lim_{n\to\infty} \lambda_i^{(n)} = \lambda_i, \qquad \lim_{n\to\infty} \left(\alpha_i^2\right)^{(n)} = \alpha_i^2, \tag{4.58}$$

and

$$\sup_{n\geq 1, i\in\mathcal{I}} \mathbf{E}\left[\left(u_{i,1}^{2+\epsilon}\right)^{(n)}\right] < \infty, \tag{4.59}$$

$$\mathbf{E}\left[v_{i,j,1}^{2+\epsilon}\right] \equiv d_{i,j} < \infty, \tag{4.60}$$

for some $\epsilon > 0$ and finite constant $d_{i,j}$, $j \in \mathcal{J}$.

From (4.51)–(4.54) and (4.58)–(4.60), we have the following heavy traffic conditions for queue j

$$\lim_{n\to\infty} \lambda_{i,j}^{(n)} = \lambda_{i,j}, \qquad \lim_{n\to\infty} \left(\alpha_{i,j}^2\right)^{(n)} = \alpha_{i,j}^2, \tag{4.61}$$

$$\lim_{n\to\infty} \mu_{e_j}^{(n)} = \mu_{e_j}, \qquad \lim_{n\to\infty} \left(\beta_{e_j}^2\right)^{(n)} = \beta_{e_j}^2, \tag{4.62}$$

and

$$\sup_{n\geq 1, i\in\mathcal{I}} \mathbf{E}\left[\left(\tilde{u}_{i,j}^{2+\epsilon}\right)^{(n)}\right] < \infty, \tag{4.63}$$

$$\mathbf{E}\left[\left(v_{e_j,1}^{2+\epsilon}\right)^{(n)}\right] = \sum_{i=1}^{I} \frac{\lambda_{i,j}}{\lambda_{e_j}} d_{i,j} < \infty, \quad \forall j \in \mathcal{J}, \tag{4.64}$$

where $\{v_{e_j,m} : m \geq 1\}$ is a sequence of i.i.d. random variables formed by the effective processing times at each server j. In addition, from (4.10), (4.11) and (4.57), we have

$$\lim_{n\to\infty} \sqrt{n}\Big(\sum_{i=1}^{I} \lambda_{i,j}^{(n)} - \mu_{e_j}\Big) = c_j. \tag{4.65}$$

Let $\hat{Q}_j^{(n)}(t)$ be the diffusion scaled queue length process for queue j. From (4.61)–(4.65) and Theorem 2.2.5, we have the following result.

Theorem 4.3.1. *$\hat{Q}_j^{(n)}(t) \xrightarrow{w} \hat{Q}_j = RBM\left(c_j,\ \sigma_j^2\right)$, as $n \to \infty$, where*

$$\begin{aligned}
\sigma_j^2 &= \sum_{i=1}^{I} \lambda_{i,j}^3 \alpha_{i,j}^2 + \mu_{e_j}^3 \beta_{e_j}^2 \\
&= \sum_{i=1}^{I} \lambda_i (\lambda_i^2 \alpha_i^2 - 1) p_{i,j}^2 + \left(\sum_{i=1}^{I} \lambda_i p_{i,j}\right)^2 \left[\sum_{i=1}^{I} \lambda_i (\mu_{i,j}^{-2} + \beta_{i,j}^2) p_{i,j}\right].
\end{aligned}$$

Let $C_{a,i}^2 = \lambda_i^2 \alpha_i^2$ and $C_{s,i,j}^2 = \mu_{i,j}^2 \beta_{i,j}^2$ denote the squared coefficients of variation of the inter-arrival times and the processing times, respectively. Even though the queue length process $Q_j^{(n)}(t)$ are not mutually independent due to the geometric arrival splitting, we can still obtain the weighted total mean as

$$\tilde{\varphi} = \sum_{j=1}^{J} \frac{\sum_{i=1}^{I} \delta_i p_{i,j}^2}{\sum_{i=1}^{I} p_{i,j}} + \sum_{j=1}^{J} \frac{\left(\sum_{i=1}^{I} \lambda_i p_{i,j}\right)^2 \left(\sum_{i=1}^{I} \theta_{i,j} p_{i,j}\right)}{\sum_{i=1}^{I} p_{i,j}}, \tag{4.66}$$

where

$$\delta_i = \lambda_i (C_{a,i}^2 - 1), \quad \theta_{i,j} = \lambda_i (C_{s,i,j}^2 + 1)/\mu_{i,j}^2.$$

If that we were looking at anything other than the mean, we would have to take the correlation between the arrival streams into account. If the arrivals follow Poisson processes and the servers are identical, (4.66) is the same as (4.19).

Define

- decision variables $\mathbf{x} = vec(P)$ and augmented variables $\mathbf{y} = [y_1, ..., y_{4J}]^T$;
- parameters $\delta = (\delta_i)_{1\times I}$, $\rho = (\rho_{i,j})_{I\times J}$ and $\theta = (\theta_{i,j})_{I\times J}$. The sign of δ_i depends on the value of the squared coefficients of variation of the inter-arrival times. For all $i \in \mathcal{I}$, $j \in \mathcal{J}$, we have $\rho_{i,j} \geq 0$ and $\theta_{i,j} \geq 0$. (By convention, $\rho_{i,j} = 0$ and $\theta_{i,j} = 0$ if $\mu_{i,j} = 0$.)

The corresponding NLP is formulated as

$$\min_{(\mathbf{x},\mathbf{y})} \quad \tilde{\varphi}(\mathbf{y}) = \sum_{j=1}^{J} \left(y_j^2\right) \left(y_{j+J}\right) \left(y_{j+2J}^{-1}\right) + \sum_{j=1}^{J} \left(y_{j+3J}\right) \left(y_{j+2J}^{-1}\right) \tag{4.67}$$

$$\text{s.t.} \quad y_j - \sum_{i=1}^{I} \lambda_i x_{(j-1)I+i} = 0, \quad \forall j \in \mathcal{J}$$

$$y_{j+J} - \sum_{i=1}^{I} \theta_{i,j} x_{(j-1)I+i} = 0, \quad \forall j \in \mathcal{J}$$

$$y_{j+2J} - \sum_{i=1}^{I} x_{(j-1)I+i} = 0, \quad \forall j \in \mathcal{J}$$

$$y_{j+3J} - \sum_{i=1}^{I} \delta_i x_{(j-1)I+i}^2 = 0, \quad \forall j \in \mathcal{J}$$

$$\sum_{i=1}^{I} \rho_{i,j} x_{(j-1)I+i} = 1, \quad \forall j \in \mathcal{J}$$

$$\sum_{j=1}^{J} x_{(j-1)I+i} = 1, \quad \forall i \in \mathcal{I}$$

$$0 \leq x_n \leq 1, \quad \forall n \in \{1, ..., IJ\}$$

$$0 \leq y_j \leq \sum_{i=1}^{I} \lambda_i, \quad 0 \leq y_{j+J} \leq \sum_{i=1}^{I} \theta_{i,j}, \quad 0 \leq y_{j+2J} \leq I,$$

$$\min\left(0, \sum_{i=1}^{I} \delta_i\right) \leq y_{j+3J} \leq \max\left(0, \sum_{i=1}^{I} \delta_i\right), \quad \forall j \in \mathcal{J}.$$

If the arrivals follow Poisson processes and the servers are identical, NLP (4.67) is the same as NLP (4.24). Although NLP (4.67) has both linear and *non-linear* constraints, it can still be solved by a generic NLP solver like KNITRO.

Except for the variable substitution constraints, NLP (4.67) has the same linear constraints as LP (3.2) with equality constraints. This implies that (1) using (4.56), the routing matrix derived from the solution to LP (3.2) can be used as a starting point of $\mathbf{x}$ to solve NLP (4.67); (2) using Proposition 4.1.1, the optimal routing matrix contains many zero elements, especially when the matrix is large.

4.4 Summary

To derive the optimal (static) random routing policy for output-queued systems, we have formulated a nonlinear programming problem which minimizes the delay in the system in heavy traffic. When inter-arrival times are exponentially distributed, the resulting optimal policy does not change if the processing times distributions are changed such that the (squared) coefficient of variation is altered by the same factor for all task types at each server. Using the optimal routing policy, most servers need only to be capable of processing a small subset of task types. This is desirable when it is costly to maintain highly flexible servers.

In the case of homogeneous server systems, the optimal routing matrix contains identical columns. Pooling the corresponding servers into a single queue can further reduce the delay in the system. To choose between full pooling and partial pooling, one can compare the diffusion limits of the total mean queue length of the system in heavy traffic.

For systems in moderate traffic, we have also formulated a nonlinear programming problem which minimizes the mean sojourn time for arrivals that follow Poisson processes. In the case of homogeneous server systems, it is hard to obtain the optimal pooling strategy analytically, since the mean queue lengths of multi-server queues can only be estimated with approximations.

Chapter 5

Applications

In this chapter, we discuss applications of the MARO related routing policies proposed in Chapter 3. First in Section 5.1, simulation studies are used to demonstrate the heavy traffic optimality properties in the systems which are equipped with homogeneous or heterogeneous servers to process a single type of tasks. In Section 5.2, we apply the MARO related policies to a server cluster environment that processes multiple types of tasks, exemplifying their applications in resource management for distributed computing systems. Finally, Section 5.3 discusses the issues of applying the MARO policy to reduce hospital waiting times.

5.1 Single-Task-Type Systems

5.1.1 Homogeneous Systems

Consider an output-queued system with J identical servers, whose service times are assumed independent, each with mean one. The arrival stream consists of one task type and follows a Poisson process with rate $\rho^* J$. As discussed in Section 3.4.1, the JSQ policy is a special case of the MARO policy when applied to this homogeneous system with a single task type. Therefore, we compare the following routing structures when the system is heavily loaded, i.e., ρ^* is close to one.

- LP−Static: According to the solution to LP (3.2), arrivals are routed to the J servers with equal probabilities $1/J$. Since the arrival stream follows a Poisson process, the system is thus equivalent to J $M/G/1$ queues in parallel.

- JSQ−flex: This is a special case of MARO−flex. When the flexibility level $q = 1$, it is JSQ. So a flexible arrival joins one of the J servers with the shortest queue length. Ties are broken randomly. A dedicated arrival is routed to one of the J servers with probability $1/J$.

- JSQ−$2/k$: This is a special case of MARO−$2/k$. An arrival joins the shorter of the two queues chosen from the J servers. The two candidate queues are picked with probabilities $1/J$ and $1/(J-1)$, respectively.

- JSQ−tree: This is a special case of MARO−tree. A flexible arrival is routed to queue j with probability given by (3.57) and joins the shorter of queues j and $j+1$, $j \in \{1, ..., J-1\}$.

- JSQ−ring: This is a variant of JSQ−tree. A flexible arrival is routed to queue j with probability $1/J$ and joins the shorter of queues j and $j+1 \pmod J$, $j \in \{1, ..., J\}$.

- Full pooling: The system becomes as an $M/G/J$ queue, so no routing is needed.

Simulation Results

Systems of three different sizes are studied, using $J = 4$, 20 and 100. The simulation results mainly focus on the steady-state mean queue length of the system, which includes both the number of tasks waiting in the queue and those in process. All statistics for the dynamic routing policies are at 95 percent confidence level, with accuracies no worse than $\pm 2\%$. The mean queue lengths for the $M/G/1$ and $M/G/J$ queues are calculated using the Pollaczek-Kintchine formula [29] and the Allen-Cunneen approximation [1], respectively.

Using the results, we discuss the impact of the amount of state information required for routing and the impact of service time variance.

Impact of the amount of state information Table 5.1 compares the routing structures that require different amounts of state information. The performance improvement (Imprvmnt.) is calculated using the total mean queue length of J parallel $M/M/1$ queues as a reference, since it needs no state information. All JSQ related policies are assumed to have flexibility level $q = 1$ (i.e., no dedicated arrivals).

Table 5.1: Routing structures vs. Total mean queue lengths, i.i.d. exponential service times, $\rho^* = 0.95$

Model	$J = 4$		$J = 20$		$J = 100$	
	Mean	**Imprvmnt.**	**Mean**	**Imprvmnt.**	**Mean**	**Imprvmnt.**
$J \times M/M/1$	76.00	0 %	380.00	0 %	1900.00	0 %
JSQ−tree	25.92	66 %	96.54	75 %	461.85	76 %
JSQ−ring	23.94	68 %	89.74	76 %	446.54	77 %
JSQ−$2/k$	23.72	69 %	72.03	81 %	328.23	83 %
JSQ	22.28	71 %	45.99	88 %	176.80	91 %
$M/M/J$	20.74	73 %	33.35	91 %	104.62	94 %

It is seen that the improvements for the tree and ring structures and JSQ−$2/k$ appear to be of the same order of magnitude. This is consistent with our observations in Section 3.3.4, which would suggest that these three policies are roughly equivalent in terms of giving a significant improvement. Note that our results are also consistent with the observation in [17] that when n items are placed at n servers with d choices per item, when nearest neighbours are chosen, the maximum number of items assigned to a server is a constant factor larger than a system where the d choices are made randomly. In our case where $n = J$ and $d = 2$, the tree and ring structures yield only a constant increase of the mean queue length (per queue) over JSQ−$2/k$.

When compared using the discounted amount of the required state information for routing (given by (3.58)), JSQ−tree and JSQ−$2/k$ require significantly less information than JSQ. When the number of servers increases

from 4 to 100, the discount increases from 50% to 98%. However, the difference (in terms of the total mean queue length) between JSQ−tree (or JSQ−2/k) and JSQ also becomes larger in systems with a larger number of identical servers (see also Figure 5.1). Therefore, there is a tradeoff between the system performance and the cost of acquiring more state information in homogeneous systems.

In addition to Table 5.1, Figure 5.1 also compares the JSQ related policies with an $M/M/J$ queue. Although Theorem 3.4.1 implies that under the heavy traffic condition (i.e., $\rho \to 1$), the JSQ related policies yield the same diffusion scaled queue length which is in turn also the same as that of an $M/M/J$ queue, it can be seen that when one backs off from heavy traffic, even at 95 percent load, the actual queue lengths differ in systems with a larger number of servers. New techniques ([28] for example) are needed to yield diffusion scaled limits that allow one to differentiate between various policies in finer granularity.

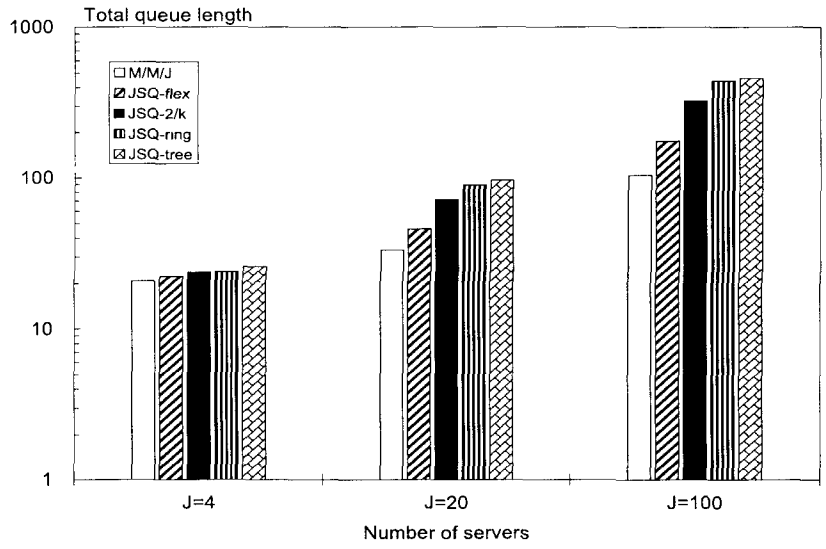

Figure 5.1: Routing structures vs. Total mean queue lengths, i.i.d. exponential service times, $\rho^* = 0.95$

The flexibility level q is another factor that characterizes the amount of state information required for routing. Figure 5.2 shows that under high

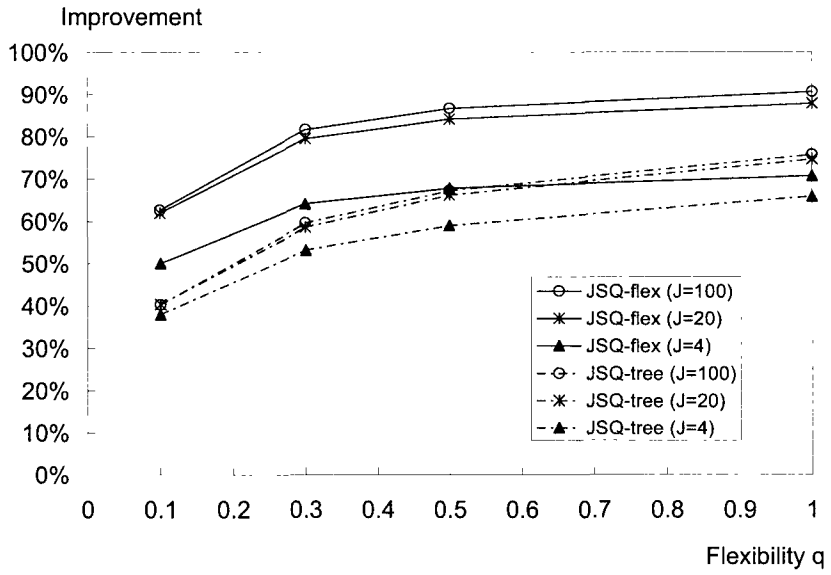

Figure 5.2: Flexibility levels vs. Improvement of total mean queue length, i.i.d. exponential service times, $\rho^* = 0.95$

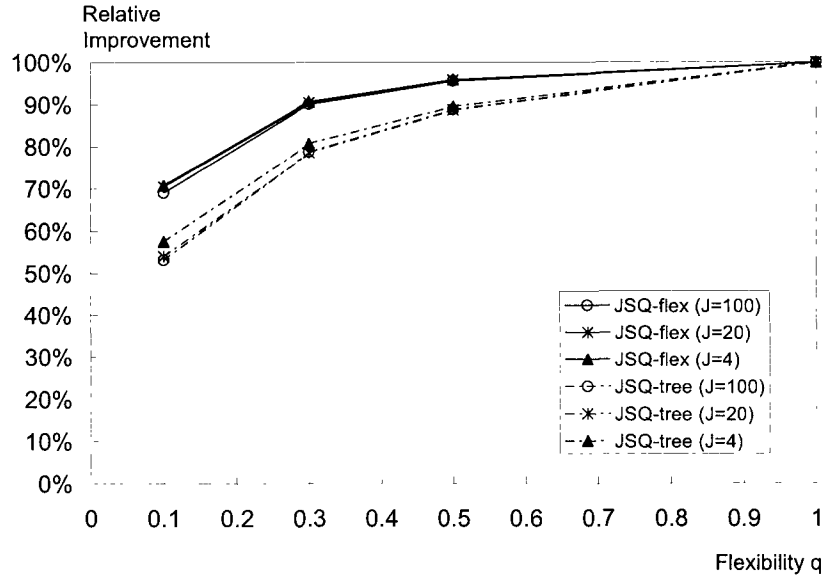

Figure 5.3: Flexibility levels vs. Relative improvement of total mean queue length, i.i.d. exponential service times, $\rho^* = 0.95$

load, there is a significant improvement for a very small level of flexibility: about 40 percent improvement for JSQ−tree and 50 to 60 percent improvement for JSQ−flex, respectively, at 10 percent flexibility. At a given level q, the improvements increase as the number of queues J increases. Figure 5.3 shows that at 30 percent flexibility, the amount of improvement achieved by JSQ−tree is about 80 percent of that with 100 percent flexibility. This relative improvement increases to 90 percent for JSQ−flex. These observations are consistent with Theorem 3.4.1, which implies that under heavy traffic, the diffusion scaled queue lengths for both policies are independent of the flexibility level q.

Impact of processing time variance To examine how processing time variance affects the system performance, we compare the total mean lengths using two more processing time distributions: Erlang−k and hyper-exponential, in addition to exponential. The squared coefficient of variation C_X^2 is set to 0.1 for Erlang−k (i.e., $k = 10$) and 10 for hyper-exponential, respectively. (By definition, $C_X^2 = 1$ for the exponential distribution.)

The results given in Tables 5.2 and 5.3, as well as in Figures 5.4–5.7, are consistent with the observations made for the exponential service times setting. In addition, it can be seen that all policies have larger improvement in systems with larger service time variance than in those with small variance. This is probably not too surprising, as it follows from the observation that when the service time variance is small, the performance is less sensitive to the policy, i.e., for small service time variance if some policy balances the load over long time scales, it is highly likely to also balance the load under shorter time scales. For example, in the extreme of constant service times, an optimal routing policy would be round robin. On the other hand, with large service time variance, load imbalances may occur over short time scales due to the variability in service times, so it becomes more desirable to be able to shift the incoming work between queues.

Table 5.2: Routing structures vs. Total mean queue lengths, i.i.d. Erlang$-k$ service times, $\rho^* = 0.95$

Model	$J = 4$		$J = 20$		$J = 100$	
	Mean	**Imprvmnt.**	**Mean**	**Imprvmnt.**	**Mean**	**Imprvmnt.**
$J \times M/E_k/1$	43.51	0 %	217.55	0 %	1087.75	0 %
JSQ$-$tree	16.08	63 %	63.58	71 %	298.45	73 %
JSQ$-$ring	14.89	66 %	57.90	73 %	291.22	73 %
JSQ$-2/k$	14.67	66 %	47.26	78 %	219.57	80 %
JSQ	14.06	68 %	36.21	83 %	156.95	86 %
$M/E_k/J$	13.12	70 %	26.89	88 %	100.29	91 %

Table 5.3: Routing structures vs. Total mean queue lengths, i.i.d. hyper-exponential service times, $\rho^* = 0.95$

Model	$J = 4$		$J = 20$		$J = 100$	
	Mean	**Imprvmnt.**	**Mean**	**Imprvmnt.**	**Mean**	**Imprvmnt.**
$J \times M/H_x/1$	400.90	0 %	2004.50	0 %	10022.50	0 %
JSQ$-$tree	113.99	72 %	368.36	82 %	1684.35	83 %
JSQ$-$ring	101.87	75 %	325.22	84 %	1627.27	84 %
JSQ$-2/k$	101.62	75 %	217.48	89 %	900.23	91 %
JSQ	96.16	76 %	108.63	95 %	218.92	98 %
$M/H_x/J$	96.95	76 %	97.94	95 %	147.92	99 %

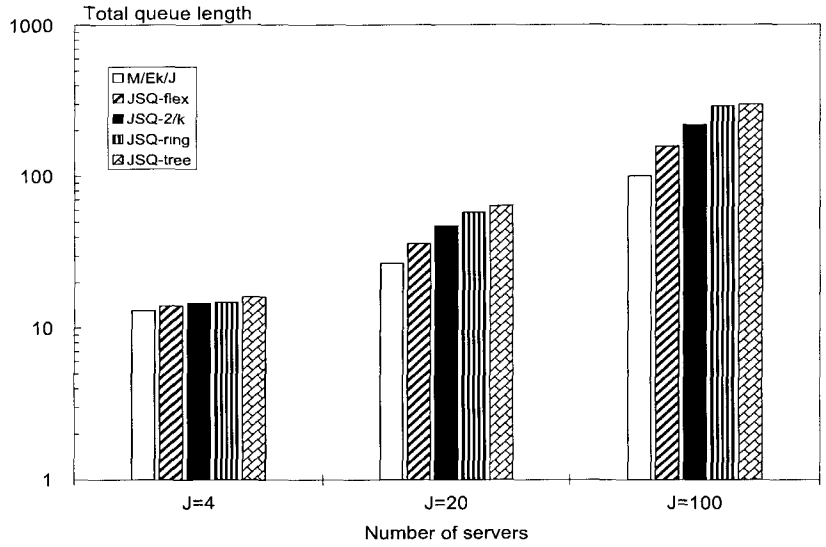

Figure 5.4: Routing structures vs. Total mean queue lengths, i.i.d. Erlang$-k$ service times, $\rho^* = 0.95$

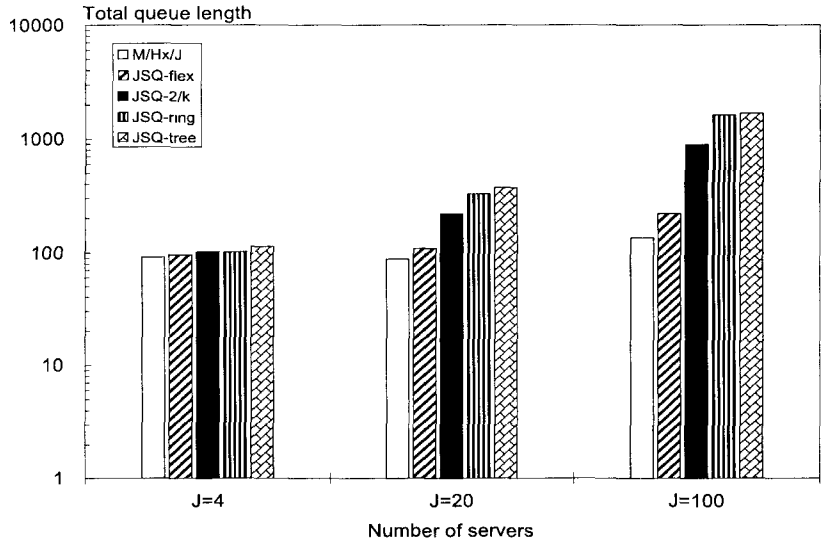

Figure 5.5: Routing structures vs. Total mean queue lengths, i.i.d. hyper-exponential service times, $\rho^* = 0.95$

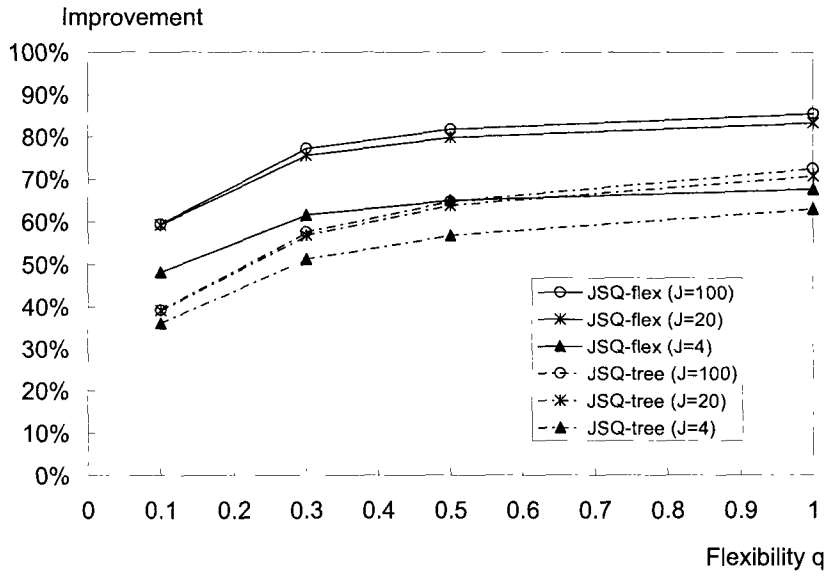

Figure 5.6: Routing structures vs. Improvement of total mean queue lengths, i.i.d. Erlang–k service times, $\rho^* = 0.95$

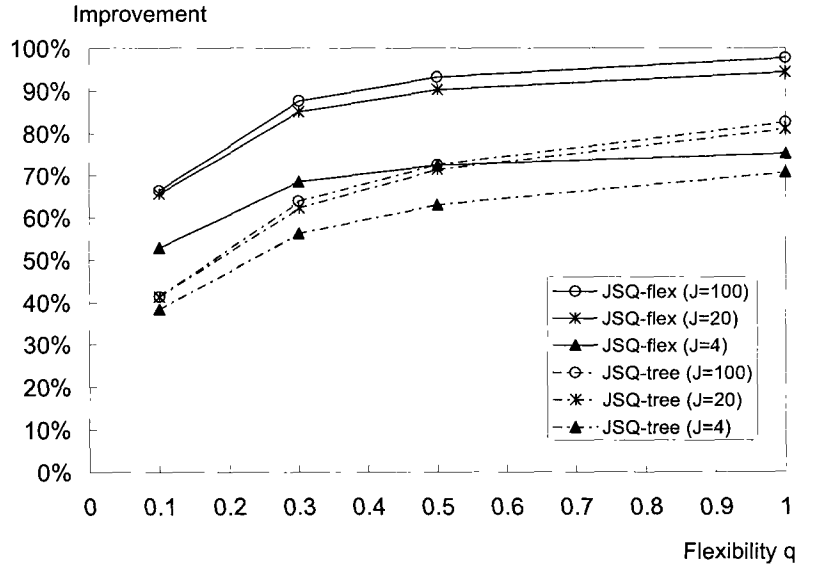

Figure 5.7: Routing structures vs. Improvement of total mean queue lengths, i.i.d. hyper-exponential service times, $\rho^* = 0.95$

5.1.2 Heterogeneous Systems

Here we study a single-task-type system with $J = 20$ heterogeneous servers. The service time distribution at queue j is exponential with rate μ_j. Let the mean service rate vector be $\mu = [1, 2, \ldots, 20]$, and the single Poisson arrival stream have rate $\tilde{\lambda} = 199.5$, so that the solution to LP (3.2) is $\rho^* = 0.95$ and $\Psi^* = (\rho^*)_{1\times J}$, a vector of J ρ^*'s.

Corresponding to the JSQ related policies given in Section 5.1.1, we compare the JSEW related policies with J different $M/M/1$ queues in parallel and one $M/M/J^h$ queue with J heterogeneous servers, respectively. The flexibility level q is set to one.

Table 5.4: Routing structures vs. Total mean queue lengths, heterogeneous servers, exponential service times, $\rho^* = 0.95$

Model	Mean	Imprvmnt.
$J^h \times M/M/1$	380.00	0 %
JSEW−tree	88.02	77 %
JSEW−ring	87.78	77 %
JSEW−$2/k$	70.66	81 %
JSEW	37.02	90 %
$M/M/J^h$	34.52	91 %

Using LP−Static, the arrivals are routed to queue j at rate $0.95\mu_j$, so the total mean queue length of J parallel $M/M/1$ queues is the same as that in the homogeneous server case. Table 5.4 shows that JSEW−tree and JSEW yield similar improvements as those seen in Table 5.1, the homogeneous server case. Actually, we know from Theorems 3.4.1 and 3.4.2 that in the case of exponential service times, both the homogeneous and the heterogeneous systems have the same reflected Brownian motion limit (under the complete resource pooling condition), so this observation is not surprising.

5.2 Grid Systems

Grid systems [23] are gaining acceptance as the preferred way to coordinate computing resources across institutional boundaries. A Grid is a generalized distributed computing system that can scale to Internet-size environments with machines distributed across multiple administrative domains. Computer clusters form the natural building blocks for grid systems by networking a set of independent machines with various computational capabilities to perform a specific set of applications. For a Grid to efficiently support a variety of applications, the resource management system (RMS) [44] is essential to its operation and is responsible for optimizing its performance metric.

One of the core functions of an RMS is to allocate resources (a machine or some service that is synthesized using a combination of machines, networks and software) and route incoming tasks to machines in a computer cluster. We propose in Chapter 3 a new routing policy, MARO, and several variants. By applying them to a real-world server cluster, we show that our proposed policies have several advantages over existing policies.

5.2.1 The Base Model

For our study, we adopt the server cluster environment used in Kontothanassis' experiment [43]. The characteristics of the machines in the cluster are summarized in Table 5.5. The tasks to be executed in the server cluster are the jobs performed by the popular BLAST [2] application suite, which biologists use to search nucleotide and protein databases in the National Center for Biotechnology Information in the United States. For example, one type of tasks called "blastn" compares a nucleotide query sequence against a nucleotide sequence database. There are five task types: some depend primarily on CPU speed, others are more I/O bound and thus sensitive to the memory and disk subsystems. The mean processing times of each task type at the six machine types are given in [43] and the variances are known to be small. Our study is to use the six types of machines and the five types of tasks to construct different server clusters and to compare the performance of the systems using different server allocation policies (including MARO). The performance metric

chosen is the steady-state mean total number of tasks in the server cluster. By Little's Law, it is then easy to calculate the average task completion time.

Table 5.5: Machine types in a BLAST server cluster

Machine	CPU	Memory	Disk
Type 1	733 MHz	256 MB	40 MB/s IDE
Type 2	525 MHz	8 GB	60 MB/s SCSI
Type 3	2.8 GHz	256 MB	40 MB/s IDE
Type 4	2.8 GHz	256 MB	60 MB/s SCSI
Type 5	2.8 GHz	1.5 GB	40 MB/s IDE
Type 6	2.8 GHz	1.5 GB	60 MB/s SCSI

Define sets $\mathcal{I} = \{1, ..., I\}$ and $\mathcal{J} = \{1, ..., J\}$. The base cluster system has $J = 6$ servers, each being a different machine type and capable of processing $I = 5$ types of tasks. The processing times are assumed independent. The local scheduling policy at each server is first-come-first-serve. The full trace of tasks (with all five types) follows a Poisson process with rate $\tilde{\lambda} = 0.877\, sec^{-1}$. Let the i-th element of the vector p^t denote the proportion of type i tasks in the full trace, here

$$p^t = (p_i^t)_{1\times I} = \left[\begin{array}{ccccc} 0.558 & 0.188 & 0.212 & 0.014 & 0.028 \end{array}\right]. \tag{5.1}$$

Let $\lambda_i = \tilde{\lambda} p_i^t$ be the mean arrival rate of type i tasks and $\mu_{i,j}$ be the mean processing rate of type i tasks at server j. We have the first-order primitives

$$\lambda = (\lambda_i)_{1\times I} = \left[\begin{array}{ccccc} 0.489 & 0.165 & 0.186 & 0.012 & 0.025 \end{array}\right], \tag{5.2}$$

and

$$\mu = (\mu_{i,j})_{I\times J} = \left[\begin{array}{cccccc} 0.385 & 0.571 & 0.556 & 0.667 & 0.588 & 1.111 \\ 0.699 & 1.111 & 1.786 & 1.923 & 3.125 & 3.333 \\ 0.435 & 0.556 & 1.111 & 1.053 & 1.667 & 1.667 \\ 0.070 & 0.069 & 0.174 & 0.174 & 0.190 & 0.200 \\ 0.023 & 0.026 & 0.069 & 0.067 & 0.070 & 0.070 \end{array}\right]. \tag{5.3}$$

To have a clearer view of the qualitative attributes of the primitives, we normalize λ and μ using the smallest element $\min_{i\in\mathcal{I},j\in\mathcal{J}}\{\mu_{i,j}\}$ and obtain

$$\hat{\lambda} = \begin{bmatrix} 21.3 & 7.2 & 8.1 & 0.5 & 1.1 \end{bmatrix},$$

and

$$\hat{\mu} = \begin{bmatrix} 16.7 & 24.8 & 24.2 & 29.0 & 25.6 & 48.3 \\ 30.4 & 48.3 & 77.7 & 83.6 & 135.9 & 144.9 \\ 18.9 & 24.2 & 48.3 & 45.8 & 72.5 & 72.5 \\ 3.0 & 3.0 & 7.6 & 7.6 & 8.3 & 8.7 \\ 1.0 & 1.1 & 3.0 & 2.9 & 3.0 & 3.0 \end{bmatrix}.$$

Firstly, it can be seen that the first three "major" task types contribute more than 95 percent of arrivals for the full task trace. Secondly, the mean processing rates of task types 4 and 5 are significantly lower on all machine types, by which we may consider these two types of tasks as "hard" jobs (even though they are a small proportion of arrivals). Thirdly, using Table 5.5 and the matrix $\hat{\mu}$, we may label machine types 1 and 2 as "slow" machines and types 5 and 6 as "fast" ones. We will study how these qualitative attributes (e.g., hardness, slowness) affect the system performance.

5.2.2 Trial Systems

For the simulation study, we apply 6 server allocation policies to 5 homogeneous/heterogeneous systems which are constructed from the base system. The policies include:

- LP−Static: a static routing policy that maximizes the system capacity in the long term according to LP (3.2), but does no short term shifting of workload. Type i tasks are routed to server j with probability $p_{i,j}$ given in (3.18).

- MinDrift(Q): see Section 3.2. The cost function chosen (for the MARO related policies as well) is of the form

$$C(Z_j) = Z_j^2, \tag{5.4}$$

  where Z_j is given in (3.1), an estimate of the unfinished processing time at server j at each arrival time t. Let $Q_{n,j}(t)$ be the number of type n

tasks at queue j at time t, then a type i arrival is dispatched to queue j satisfying

$$j \in \arg\min_{j \in \mathcal{J}} \left\{ \frac{\sum_{n \in \mathcal{I}} \mu_{n,j}^{-1} Q_{n,j}(t)}{\mu_{i,j}} \right\}. \tag{5.5}$$

Ties are broken randomly with equal probabilities.

- MARO−flex: see Section 3.3.3. Assume a constant flexibility level q for all task types. When $q = 1$, it is the MARO policy (see Section 3.3.1).
- MARO−2/k: see Section 3.3.2.
- MARO−tree: see Section 3.3.4. The routing probabilities for the flexible arrivals are determined using LP (3.46).
- FCFS: a standard first-come-first-serve policy. It is used in the multi-server-single-queue systems where all tasks are waiting at one queue and the task at the head of the queue is dispatched to the first idle server. If more than one server is idle at the time of an arrival, ties are broken randomly with equal probabilities.

Note that given the cost function properties in Assumption 2.3.1, the routing decisions made based on (5.5) will not change even if the cost function is different from (5.4). This can be seen as follows. Suppose there are two cost functions $C_1(Z_j)$ and $C_2(Z_j)$ which both satisfy the assumption. Given two queues j_1 and j_2, if

$$\frac{C_1'(Z_{j_1})}{\mu_{i,j_1}} < \frac{C_1'(Z_{j_2})}{\mu_{i,j_2}},$$

then we have

$$\frac{C_2'(Z_{j_1})}{\mu_{i,j_1}} < \frac{C_2'(Z_{j_2})}{\mu_{i,j_2}},$$

if in addition $\mu_{i,j_1} = \mu_{i,j_2}$, since the first derivatives $C_1'(Z_j)$ and $C_2'(Z_j)$ are both strictly increasing in Z_j. Even if $\mu_{i,j_1} \neq \mu_{i,j_2}$, the definition of fixed point (2.15) suggests that the above relations still hold in heavy traffic (see Theorem 2.3.2). Therefore, the qualitative comparison results obtained in the following

simulation studies are not sensitive to the specific form of a cost function which satisfies Assumption 2.3.1.

There are two categories of systems constructed. Systems A and B are homogeneous systems, in which all machines in the server cluster are of one type. Systems C, D and E are heterogeneous systems, in which the server cluster consists of six different machine types. The applied server allocation policies are summarized in Table 5.6.

Table 5.6: Trial systems and the applied server allocation policies

	System A	System B	System C	System D	System E
LP−Static	√		√	√	
MinDrift(Q)	√		√	√	
MARO−flex	√		√	√	
MARO−2/k	√		√	√	
MARO−tree	√		√	√	
FCFS		√			√

Homogeneous Systems

System A This system has $J = 6$ identical parallel servers, each capable of processing $I = 5$ types of tasks. All servers are chosen to be machine type 2 so the results can be compared with those in [43]. From (5.2) and (5.3), we have the first-order primitives ${}^{A}\lambda = \lambda$ and

$${}^{A}\mu = \begin{bmatrix} \mu_{i,2} & \mu_{i,2} & \mu_{i,2} & \mu_{i,2} & \mu_{i,2} & \mu_{i,2} \end{bmatrix}_{i\in\mathcal{I}}.$$

As has been analyzed in Section 3.4.2, if the full matrix Ψ^* (3.60) is to be used in this homogeneous system, the LP-Static policy routes a task to each of the servers with equal probabilities. Thus the system behaves as J parallel $M/G/1$ queues with each arrival stream a Poisson process with rate $\tilde{\lambda}/J$.

The MinDrift(Q) policy reduces to the "join the shortest expected waiting time" (JSEW) policy, i.e., a task is dispatched to server j satisfying

$$j \in \arg\min_{j\in\mathcal{J}} \left\{ \sum_{n\in\mathcal{I}} \mu_{n,j}^{-1} Q_{n,j}(t) \right\}.$$

It is also equivalent to the "minimum completion time" (MCT) policy, which is seen in existing distributed computer systems [24].

MARO will be the same as MinDrift(Q) if the full matrix Ψ^* is used. To study the performance of the MARO related policies using less state information for routing, we will use a perturbed Ψ^* instead, which has the maximum number of zero elements (see Proposition 3.3.1).

System B This is a multi-server-single-queue system equipped with the FCFS dispatching policy, which was also used in [43]. It is included in our study as a reference for comparison. Given the processing times at all J servers are assumed i.i.d., System B is denoted as an $M^n/G/J$ queue with multiple types of arrivals. The first-order primitives are ${}^B\lambda = {}^A\lambda$ and ${}^B\mu = {}^A\mu$.

Heterogeneous Systems

System C This system has $J = 6$ heterogeneous parallel servers, each capable of processing $I = 5$ types of tasks. The first-order primitives are ${}^C\mu = \mu$ and

$$ {}^C\lambda = \lambda \times 1.89 = \left[\begin{array}{ccccc} 0.926 & 0.313 & 0.352 & 0.023 & 0.047 \end{array} \right], $$

so that the load of System C is the same as that of System A.

System D This is an expansion of System C, which has $J = 30$ parallel servers, each being capable of processing $I = 5$ types of tasks. The servers are partitioned into 6 groups, servers within a group are identical. The number of servers in each group and the corresponding mean processing rates are shown in Table 5.7. Servers in groups 1 and 2 are "slow" machines and those in groups 5 and 6 are "fast" machines. Note that these groups do not necessarily correspond to geographic locations. Depending on applications, servers of different machine types may be grouped together at one location. However, since permutation of the columns of the matrix μ does not affect the solution of LP (3.2), the results obtained in the following simulation studies apply to different groupings of the servers.

The mean arrival rate vector is

$$ {}^D\lambda = \lambda \times 9.60 = \left[\begin{array}{ccccc} 4.694 & 1.584 & 1.785 & 0.115 & 0.240 \end{array} \right], $$

Table 5.7: Mean processing rate matrix μ of System D

Machine type	slow		medium		fast	
Group No.	1	2	3	4	5	6
# of servers	2	6	7	7	4	4
Task type $i \in \mathcal{I}$	$\mu_{i,1}$	$\mu_{i,2}$	$\mu_{i,3}$	$\mu_{i,4}$	$\mu_{i,5}$	$\mu_{i,6}$

so that the load of System D is the same as that of Systems A and C.

System E Similar to System B, this is also a multi-server-single-queue system equipped with the FCFS dispatching policy. We denote System E as an $M^n/G/J^h$ queue given the processing time distributions at the J servers are different. There are two settings of the first-order primitives for this system. One is ${}^{E_1}\lambda = {}^{C}\lambda$ and ${}^{E_1}\mu = {}^{C}\mu$. The other is ${}^{E_2}\lambda = {}^{D}\lambda$ and ${}^{E_2}\mu = {}^{D}\mu$.

5.2.3 Main Results

Our main results are divided into two parts. The first part includes the optimal solutions of the resource allocation LP (3.2) for Systems A, C and D. The results are to be used in the MARO related policies. Note that the routing probabilities derived from the LP do not vary with the system loads (since the required load is obtained by multiplying the mean arrival rate vector with a positive factor while keeping the mean processing rate matrix unchanged). The second part includes simulation results comparing the server allocation policies and discusses the impact of the amount of state information required for decision making (which is indicated by flexibility level and the set of candidate queues), the impact of server utilization (and system load) and the impact of processing time variance.

Solutions of the Resource Allocation LP

System A For System A, LP (3.2) has multiple solutions Ψ^* with ${}^{A}\rho^* = 0.412$. Constraint (3.4) is active at the optimum, which means no server is

under-utilized. One solution Ψ^* is

$$ {}^{A}\Psi_1^* = \begin{bmatrix} 0.142 & 0.142 & 0.142 & 0.142 & 0.142 & 0.142 \\ 0.025 & 0.025 & 0.025 & 0.025 & 0.025 & 0.025 \\ 0.056 & 0.056 & 0.056 & 0.056 & 0.056 & 0.056 \\ 0.029 & 0.029 & 0.029 & 0.029 & 0.029 & 0.029 \\ 0.160 & 0.160 & 0.160 & 0.160 & 0.160 & 0.160 \end{bmatrix}, \tag{5.6} $$

which consists of repetitions of the first column, so the graph $\mathcal{G}_\Psi$ associated with ${}^{A}\Psi_1^*$ contains rings. The corresponding static routing probability matrix is

$$ {}^{A}P_1^d = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}. $$

As has been analyzed in Section 3.4.2, the elements of ${}^{A}\Psi_1^*$ are $\psi_{i,j} = \lambda_i/(J\mu_i)$. Therefore, System A represents a case where the loads of the fast arrivals (type 1 tasks) and of the "hard" tasks (type 5) are significantly higher than those for the other task types.

To allow the LP-based dynamic routing policies to use less state information for routing, we may perturb ${}^{A}\Psi_1^*$ along the arcs of the rings in $\mathcal{G}_\Psi$ until the number of positive elements is reduced to $(I + J - 1)$. Thus another optimal solution to LP (3.2) is obtained as

$$ {}^{A}\Psi_2^* = \begin{bmatrix} 0.276 & & & 0.076 & 0.262 & 0.238 \\ & & & & 0.150 & \\ & & & 0.336 & & \\ & & & & & 0.174 \\ 0.136 & 0.412 & 0.412 & & & \end{bmatrix}, \tag{5.7} $$

while the objective function keeps the same optimal value ${}^{A}\rho^* = 0.412$. The associated static routing probability matrix is

$$ {}^{A}P_2^d = \begin{bmatrix} 0.324 & & & 0.089 & 0.308 & 0.279 \\ & & & & 1.000 & \\ & & & 1.000 & & \\ & & & & & 1.000 \\ 0.142 & 0.429 & 0.429 & & & \end{bmatrix}. $$

Compared with ${}^{A}\Psi_1^*$ in (5.6), the structure of ${}^{A}\Psi_2^*$ allows task types 1 and 5 (whose loads are significantly higher than the others') to spread incoming workload over as many queues as possible, while keeping the maximum number of zero elements in Ψ^*.

The LP-based policies hence have two choices of Ψ^*, shown in (5.6) and (5.7). Since MARO is the same as MinDrift(Q) if (5.6) is used, we will call its variants as MinDrift related policies. For example, MinDrift−flex with flexibility level $q = 1$ is equivalent to the original MinDrift(Q) (see Table 5.10). To differentiate LP-Static in these two choices, the names MinDrift−Static and MARO−Static are used, respectively (see Table 5.11).

System C For System C, LP (3.2) has a unique solution, given by

$$ {}^{C}\Psi^* = \begin{bmatrix} 0.412 & 0.412 & & 0.110 & & 0.412 \\ & & & & 0.100 & \\ & & & & 0.211 & \\ & & & 0.020 & 0.101 & \\ & & 0.412 & 0.282 & & \end{bmatrix}. \tag{5.8} $$

and ${}^{C}\rho^* = 0.412$ which is the same as that for System A. Constraint (3.4) is active at the optimum, which means no server is under-utilized. The number of positive elements in the matrix ${}^{C}\Psi^*$ is $N_p = 10 = (I + J - 1)$, so the graph $\mathcal{G}_\Psi$ associated with ${}^{C}\Psi^*$ is a tree. The corresponding static routing probability matrix is

$$ {}^{C}P^d = \begin{bmatrix} 0.171 & 0.254 & & 0.080 & & 0.495 \\ & & & & 1.000 & \\ & & & & 1.000 & \\ & & & 0.155 & 0.845 & \\ & & 0.601 & 0.399 & & \end{bmatrix}. $$

From the last two rows of ${}^{C}P^d$, we can see that for static server allocation, the optimal solution assigns the "hard" tasks (types 4 and 5 tasks) to servers with above-average processing rates (servers 3, 4 and 5), but not necessarily to the "fastest" (server 6). At the same time, more than 40 percent of the fast arrivals (type 1 tasks) are routed to the relatively "slow" servers (server 1 being the slowest). Additionally, while server 6 has the fastest processing rate

for type 2 tasks, none of its effort is allocated to them. Instead, type 2 tasks are assigned to server 5, which has the second fastest processing rate for them. These are not obvious to deduce without the aid of the allocation LP (3.2).

For MARO−tree, the routing probabilities of the flexible arrivals are

$$ {}^{C}P^{f} = \begin{bmatrix} 0.343 & 0.158 & & 0.499 & \\ & & & & 1.000 \\ & & & & 1.000 \\ & & & 1.000 & \\ & & 1.000 & & \end{bmatrix}, $$

where the first row is obtained from LP (3.46) with $q = 1$. When the flexibility level q changes, the first row changes accordingly. For example with $q = 0.5$, we have

$$ {}^{C}(p^{f}_{1,j})_{1\times(J-1)}\Big|_{q=0.5} = \begin{bmatrix} 0.298 & 0.160 & 0.000 & 0.542 & 0.000 \end{bmatrix}. $$

System D For System D, LP (3.2) has multiple solutions for the matrix Ψ^*, each yielding ${}^{D}\rho^* = 0.412$, the same as those for Systems A and C. Constraint (3.4) is active at the optimum, which means no server is under-utilized. One solution of ${}^{D}\Psi^*$ and the corresponding static probability matrix ${}^{D}P^{d}$ are shown in Tables 5.8 and 5.9, respectively. Servers within the same group are assigned the same non-zero values $\psi^*_{i,j}$ and $p^d_{i,j}$. The number of non-zero elements in the matrix ${}^{D}\Psi^*$ is $N_p = 52 > (I + J - 1)$, so the graph $\mathcal{G}_\Psi$ associated with ${}^{D}\Psi^*$ contains rings. We will use Table 5.9 for the MARO related policies in the simulation studies, since the discounted amount of required state information is still very significant (see Table 5.10). For MARO−tree, the routing probabilities for the flexible arrivals are obtained using Table 5.9 and LP (3.46).

It is interesting to note that although System A might be seen as a subsystem of System D (since A has both the same number and the same type of servers as group 2 in D, except that the mean arrival rate vectors are differentiated by a constant factor, ${}^{D}\lambda = {}^{A}\lambda \times 9.6$), the server allocations are quite different. This shows that one must take into account the relative values of the processing rates at different servers. Consequently it suggests

Table 5.8: Optimal solution matrix Ψ^* of System D

Machine type	slow		medium		fast	
Group No.	1	2	3	4	5	6
# of servers	2	6	7	7	4	4
Task type 1	0.412	0.412		0.242		0.412
Task type 2					0.127	
Task type 3					0.267	
Task type 4				0.083	0.018	
Task type 5			0.412	0.087		

Table 5.9: Static routing probability matrix P^d of System D

Machine type	slow		medium		fast	
Group No.	1	2	3	4	5	6
# of servers	2	6	7	7	4	4
Task type 1	0.035	0.050		0.034		0.098
Task type 2					0.250	
Task type 3					0.250	
Task type 4				0.126	0.030	
Task type 5			0.119	0.024		

that when the servers in a system change, the optimal routing policy needs to change accordingly. Again, such changes would be difficult to deduce without the aid of the allocation LP (3.2).

Finally, for Systems A, C and D, the discounted amounts of required state information for the MARO−flex, MARO$-2/k$ and MARO−tree policies are calculated using (3.19), (3.16) and (3.37), respectively. The results are summarized in Table 5.10. It can be seen that in a relatively large system like System D, the amount of state information required for making routing decisions is reduced significantly, even if we do not use the matrix Ψ^* which has the maximum number of zeros. For the MinDrift related policies which use the full matrix ${}^C\Psi^*$ in the homogeneous case (System A), the corresponding discounted amount of information is also attached.

Table 5.10: Discounted amount of required state information

	MARO–flex			MARO–$2/k$	MARO–tree		
	$q=1$	$q=0.5$	$q=0.3$		$q=1$	$q=0.5$	$q=0.3$
System A	61%	81%	88%	81%	81%	90%	94%
System C	61%	81%	88%	81%	81%	90%	94%
System D	58%	79%	87%	94%	94%	97%	98%
	MinDrift–flex			MinDrift–$2/k$	MinDrift–tree		
	$q=1$	$q=0.5$	$q=0.3$		$q=1$	$q=0.5$	$q=0.3$
System A	0%	50%	70%	68%	68%	84%	90%

Simulation Results

The simulation results mainly focus on the steady-state mean queue length. All statistics for the dynamic routing policies are at 95 percent confidence level, with the accuracy calculated as the ratio of the half width of the confidence interval to the mean value.

When equipped with the LP–Static policy, either a homogeneous or a heterogeneous system is equivalent to J $M/G/1$ queues in parallel. Given the probability $p^d_{i,j}$ defined in (3.18) (with which type i tasks are routed to queue j) and the processing times at server j with mean $\mu_{i,j}^{-1}$ and variance $\beta^2_{i,j}$, the mean queue length Q_j of each queue j can be calculated using the Pollaczek-Kintchine formula as described in Section 4.3.1.

Since System B is equivalent to an $M/G/J$ queue with multiple arrival types, its mean queue length can be approximated using the Allen-Cunneen formula

$$Q \approx \rho_e + \frac{C(J,\rho_e)\,\rho_e(1+\mu_e^2\beta_e^2)}{2(J-\rho_e)},$$

where $C(\cdot,\cdot)$ is the Erlang-C formula and $\rho_e = \tilde{\lambda}/\mu_e$ is the effective offered load, with $\tilde{\lambda}$, μ_e and β_e^2 as defined in (4.34)–(4.36).

For exponential processing time distributions, Tables 5.11 – 5.16 compare the performance improvements that can be achieved using different routing policies. The improvements, $Imprvmnt_1$ and $Imprvmnt_2$, are calculated

using the LP−Static policy and MinDrift(Q) as a reference, respectively. A negative improvement means a policy being outperformed by the reference. The performance degradation of a policy is compared with the corresponding discounted amount of required state information shown in Table 5.10.

Impact of the amount of state information The amount of state information required in a dynamic routing policy is characterized by two factors: (1) whether the policy is based on LP (3.2), e.g., MinDrift(Q) is not but MARO is; (2) the flexibility level adopted by the policy. From the results, several observations can be made.

Firstly, in the homogeneous case shown in Table 5.11, the MARO related dynamic policies using the perturbed matrix Ψ^* (5.7) perform better than their counterparts using the full matrix (5.6) (except that MARO is outperformed by MinDrift(Q)). However, as the system load increases, we see the opposite in Table 5.12. As has been analyzed in Section 3.4.2, such performance results from unbalanced workload between homogeneous servers, due to the limited choices enforced by the MARO related dynamic policies. This can be seen from Table 5.13, where the MARO related policies yield larger standard deviations of the server utilizations than the MinDrift related ones. This unbalancing may have an adverse effect on system performance, especially under high load. Therefore, for homogeneous systems, we recommend the MARO related policies use the full matrix Ψ^*. To reduce the amount of state information required for routing, we recommend using MinDrift$-2/k$.

Secondly, in the heterogeneous cases shown in Tables 5.14, 5.15 and 5.16, MinDrift(Q) performs worse than MARO, especially when the system is heavily loaded. Although it has been proved [63] that MinDrift(Q) will route the tasks to the corresponding servers according to the LP-derived probabilities (3.18) under the heavy traffic condition, our results suggest that as one backs off from heavy traffic, MinDrift(Q) routes *all* types of tasks to the slowest server with non-zero probabilities, which can result in problematic routing choices. This phenomenon can also be seen in Table 5.17, where MinDrift(Q) makes each server's utilization higher than the optimal value 0.41 (the average being above 0.52 in both Systems C and D). It has been observed that the

Table 5.11: Total mean queue lengths (Systems A and B, $\rho^* = 0.41$), exponential processing times

Policies	q	**Mean**	**Accuracy**	**Imprvmnt$_1$**	**Imprvmnt$_2$**	*Discnt*
MinDrift−Static	–	12.86	–	0 %	–	–
MinDrift−flex	1	2.58	± 0.4 %	80 %	0 %	0 %
	0.5	6.32	± 0.8 %	51 %	−145 %	50 %
	0.3	8.45	± 0.9 %	34 %	−228 %	70 %
MinDrift−$2/k$	–	3.98	± 0.6 %	69 %	−54 %	68 %
MinDrift−tree	1	4.56	± 0.7 %	65 %	−77 %	68 %
	0.5	7.55	± 1.0 %	41 %	−193 %	84 %
	0.3	9.25	± 1.1 %	28 %	−259 %	90 %
$M^n/G/J$	–	2.54	–	80 %	–	–
MARO−Static	–	5.99	–	0 %	–	–
MARO−flex	1	3.06	± 0.3 %	49 %	−19 %	61 %
	0.5	4.43	± 0.5 %	26 %	−72 %	81 %
	0.3	5.06	± 0.6 %	16 %	−96 %	88 %
MARO−$2/k$	–	3.36	± 0.4 %	34 %	−54 %	81 %
MARO−tree	1	3.42	± 0.4 %	43 %	−33 %	81 %
	0.5	4.38	± 0.6 %	27 %	−70 %	90 %
	0.3	4.90	± 0.8 %	18 %	−90 %	94 %

Table 5.12: Total mean queue lengths (Systems A and B, $\rho^* = 0.95$), exponential processing times

Policies	q	**Mean**	**Accuracy**	**Imprvmnt$_1$**	**Imprvmnt$_2$**	*Discnt*
MinDrift−Static	–	660.18	–	0 %	–	–
MinDrift(Q)−flex	1	101.41	± 3.1 %	85 %	0 %	0 %
	0.5	119.76	± 2.6 %	82 %	−18 %	50 %
	0.3	147.09	± 2.2 %	78 %	−45 %	70 %
MinDrift−$2/k$	–	113.89	± 2.8 %	83 %	−12 %	68 %
MinDrift−tree	1	150.61	± 2.1 %	77 %	−49 %	68 %
	0.5	195.91	± 2.0 %	71 %	−93 %	84 %
	0.3	249.28	± 1.9 %	63 %	−146 %	90 %
$M^n/G/J$	–	95.57	–	86 %	–	–
MARO−Static	–	237.16	–	0 %	–	–
MARO(Q)−flex	1	181.21	± 2.1 %	24 %	−79 %	61 %
	0.5	165.70	± 2.3 %	30 %	−63 %	81 %
	0.3	152.74	± 2.2 %	36 %	−51 %	88 %
MARO−$2/k$	–	163.90	± 2.3 %	31 %	−62 %	81 %
MARO−tree	1	229.47	± 1.8 %	3 %	−126 %	81 %
	0.5	233.97	± 1.9 %	1 %	−131 %	90 %
	0.3	225.23	± 1.9 %	5 %	−122 %	94 %

Table 5.13: Server utilizations (Systems A), exponential processing times

Policies	$\rho^* = 0.41$		$\rho^* = 0.95$	
	Mean	**Std**	**Mean**	**Std**
MinDrift−Static	0.412	–	0.951	–
MinDrift(Q)	0.412	± 0.002	0.951	± 0.001
MinDrift−$2/k$	0.412	± 0.012	0.950	± 0.002
MinDrift−tree ($q = 1$)	0.412	± 0.083	0.951	± 0.024
MARO−Static	0.412	–	0.951	–
MARO	0.412	± 0.071	0.950	± 0.068
MARO−$2/k$	0.412	± 0.023	0.950	± 0.059
MARO−tree ($q = 1$)	0.412	± 0.088	0.951	± 0.059

mean utilization of the slowest server is as high as 0.65 in System C and 0.68 in System D, a direct result of an inefficient assignment of tasks.

Thirdly, based on the same solution of LP (3.2), the dynamic policies outperform the static one in both homogeneous and heterogeneous systems. The improvement increases as the system size grows, although such improvement is obtained at the cost of acquiring state information for each routing decision. On the other hand, further limiting the acquired state information as MARO−$2/k$ and MARO−tree do will result in performance worse than that achieved by MARO. For the heterogeneous systems, the performance degradation becomes larger when the system size grows, as MARO−$2/k$ and MARO−tree yield larger discounts of required state information (see Table 5.10). This suggests that in designing the server allocation policies, designers should evaluate the tradeoffs between the system performance and the cost of acquiring system information before choosing one of the MARO-related policies that we have proposed.

Table 5.14: Total mean queue lengths (Systems C and E_1, $\rho^* = 0.41$), exponential processing times

Policies	q	**Mean**	**Accuracy**	**Imprvmnt$_1$**	**Imprvmnt$_2$**	*Discnt*
LP–Static	–	5.15	–	0 %	–	–
MinDrift(Q)	–	3.50	± 0.4 %	32 %	0 %	0 %
MARO–flex	1	3.45	± 0.4 %	33 %	1 %	61 %
	0.5	4.08	± 0.5 %	21 %	−17 %	81 %
	0.3	4.44	± 0.5 %	14 %	−27 %	88 %
MARO–2/k	–	3.50	± 0.4 %	32 %	0 %	81 %
MARO–tree	1	3.74	± 0.4 %	27 %	−7 %	81 %
	0.5	4.26	± 0.5 %	17 %	−22 %	90 %
	0.3	4.57	± 0.5 %	11 %	−31 %	94 %
$M^n/G/J^h$	–	3.87	± 0.7 %	25 %	–	–

Table 5.15: Total mean queue lengths (Systems D and E_2, $\rho^* = 0.41$), exponential processing times

Policies	q	**Mean**	**Accuracy**	**Imprvmnt$_1$**	**Imprvmnt$_2$**	*Discnt*
LP–Static	–	24.18	–	0 %	–	–
MinDrift(Q)	–	15.68	± 0.2%	35%	0%	0%
MARO–flex	1	13.00	± 0.2%	46%	17%	58 %
	0.5	17.83	± 0.4%	26%	−14%	79 %
	0.3	20.06	± 0.4%	17%	−28%	87 %
MARO–2/k	–	14.39	± 0.3%	40%	8%	94 %
MARO–tree	1	15.68	± 0.4%	35%	0%	94 %
	0.5	18.67	± 0.4%	22%	−19%	97 %
	0.3	20.50	± 0.5%	15%	−31%	98 %
$M^n/G/J^h$	–	17.70	± 0.3 %	27%	–	–

Table 5.16: Total mean queue lengths (Systems C, D and E, $\rho^* = 0.95$), exponential processing times

Policies	q	System C		System D	
		Mean	Accuracy	Mean	Accuracy
LP−Static	–	174.97	–	827.11	–
MinDrift(Q)	–	153.97	± 1.8 %	220.50	± 1.8%
MARO−flex	1	76.67	± 1.9 %	208.08	± 1.9 %
	0.5	85.70	± 1.6 %	199.76	± 1.6 %
	0.3	94.80	± 1.5 %	204.49	± 1.2 %
MARO−$2/k$	–	89.21	± 1.7 %	252.99	± 1.3 %
MARO−tree	1	95.87	± 1.6 %	235.18	± 1.2 %
	0.5	105.13	± 1.8 %	280.43	± 1.4 %
	0.3	113.24	± 1.3 %	325.48	± 1.2 %
$M^n/G/J^h$	–	49,938.41	± 0.7 %	236,174.33	± 0.3 %

Fourthly, when comparing a single-server-parallel-queue system with a multi-server-single-queue system (both processing multiple types of tasks), it is not surprising to see that for homogeneous systems, $M^n/G/J$ is a better choice, while for heterogeneous systems, a fully-pooled $M^n/G/J^h$ queue is not good, especially when the system load is high. The reason is that using FCFS for local scheduling, "hard" tasks (which require considerably longer processing times) will more likely block the tasks arriving afterwards and result in longer queue length. Wu showed in [71] that if both the system load and task size variance are high, the local scheduling policy had a bigger impact on the system performance than the routing policy did. Preemptive policies were proposed to replace the non-preemptive FCFS policy. Note that the output-queued systems studied in this thesis do allow preemption for tasks of different types. How to compare the performance of such a system equipped with the MARO policy with that of a fully-pooled $M^n/G/J^h$ queue, both using the same preemptive local scheduling policy, remains to be studied.

Impact of server utilization and system load As pointed out in Section 3.3.1, the long-run server utilizations have a big impact on the system performance. So we compare the dynamic policies with respect to the average and standard deviation of the utilizations of all servers in Systems A, C and D. In addition to Table 5.13, Table 5.17 shows that in heterogeneous systems, the policies based on LP (3.2) (i.e., LP−Static, MARO, MARO−$2/k$, MARO−tree) yield close-to-optimum server utilizations, while MinDrift(Q) does not. Consequently, for heterogeneous systems with 100 percent flexibility, MinDrift(Q) is outperformed by the LP-based dynamic policies in most cases, which has been shown in Tables 5.14 and 5.15 for systems in medium load, and in Table 5.16 for systems in high load. In some cases, as shown in [36], MinDrift(Q) performs even worse than the static policy, LP-Static.

Table 5.17: Server utilizations (Systems C and D, $\rho^* = 0.41$), exponential processing times

Policies	System C		System D	
	Mean	Std	Mean	Std
MinDrift(Q)	0.527	± 0.079	0.522	± 0.075
LP−Static	0.412	–	0.412	–
MARO	0.427	± 0.087	0.426	± 0.103
MARO−$2/k$	0.412	± 0.032	0.408	± 0.024
MARO−tree ($q = 1$)	0.406	± 0.125	0.412	± 0.069

Associated with server utilizations is the system load, which is indicated by the optimal solution ρ^*. The impact of system load is twofold. One is the impact on the absolute improvement that can be achieved by the dynamic policies over the static policy. The other is the impact on the relative improvement that can be achieved at different flexibility levels.

When the system load increases, the improvement over the static policy increases as well. Figure 5.8 illustrates the changes in the absolute improve-

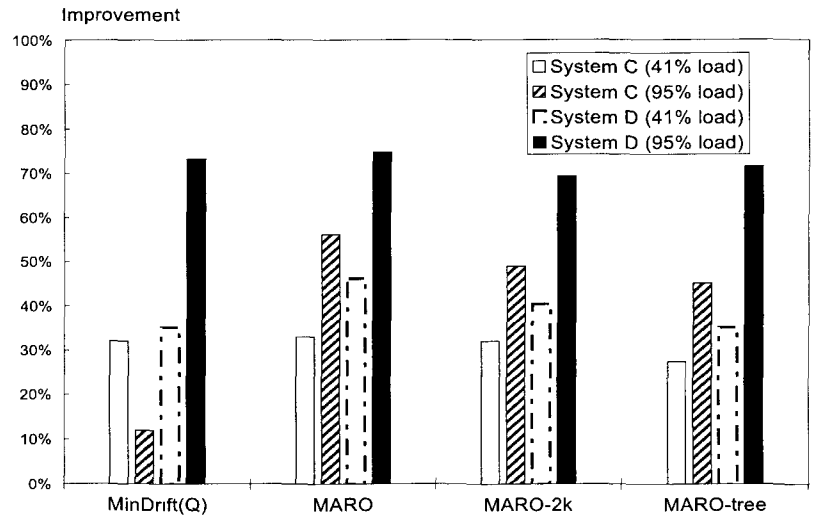

Figure 5.8: System load vs. Improvement of total mean queue length, Systems C and D, exponential processing times

ment for the heterogeneous systems with two different system loads. For example, when the load of System C increases from 41% to 95%, the improvement achieved by MARO increases from 33% to 56%.

To illustrate the impact of system load on the relative improvement of the total mean queue length, we use the heterogeneous systems equipped with the MARO–flex policy. The relative improvement is the ratio of the improvement achieved at a flexibility level q over that at 100 percent flexibility, both being calculated using the static policy as a reference. Figure 5.9 shows the relative improvement at four different flexibility levels and at two different load levels for systems with exponential processing times.

It can be seen that when System C is 41% loaded, the improvement achieved with 30 percent flexibility is 42 percent of that with 100 percent flexibility. At the same flexibility levels, the relative improvement increases to 82 percent, when the system load increases. For System D, which has a larger system size, the increase in the relative improvement goes from 37 percent to close to 100 percent. The results indicate that when the system is in heavy traffic, a small amount of flexibility can yield significant improvement in system performance. In fact, we have been able to show in Theorem 3.3.2 that in the

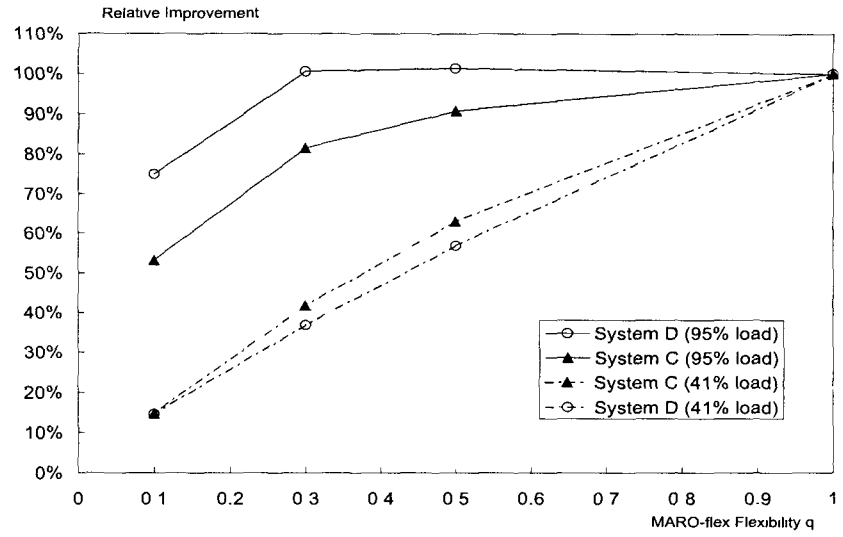

Figure 5.9: Flexibility levels vs. Relative improvement of total mean queue length, MARO−flex, exponential processing times

heavy traffic asymptotic regime, the performance improvement given by any small amount of flexibility is close to that given by 100 percent flexibility.

Impact of processing time variance To examine how processing time variance affects the system performance, we compare the total mean lengths using two more processing time distributions: deterministic and hyper-exponential, in addition to the exponential one. For the hyper-exponential case, the squared coefficient of variation $C^2_{X,i,j}$ is set to 10 for all task types i at each server j. (By definition, $C^2_X = 0$ for deterministic processing times.)

Figure 5.10 shows that when the processing time variances increase, the dynamic routing policies achieve greater improvements over the static policy, although larger processing time variances also result in longer mean total queue length (hence longer sojourn time in the system). Observations made from Tables 5.14 and 5.15 still hold for different processing time distributions with smaller or larger variances. The MinDrift(Q) policy not based on LP (3.2) performs even worse when the processing time variances are smaller. In the deterministic case, MinDrift(Q) is outperformed by the static policy, LP−Static.

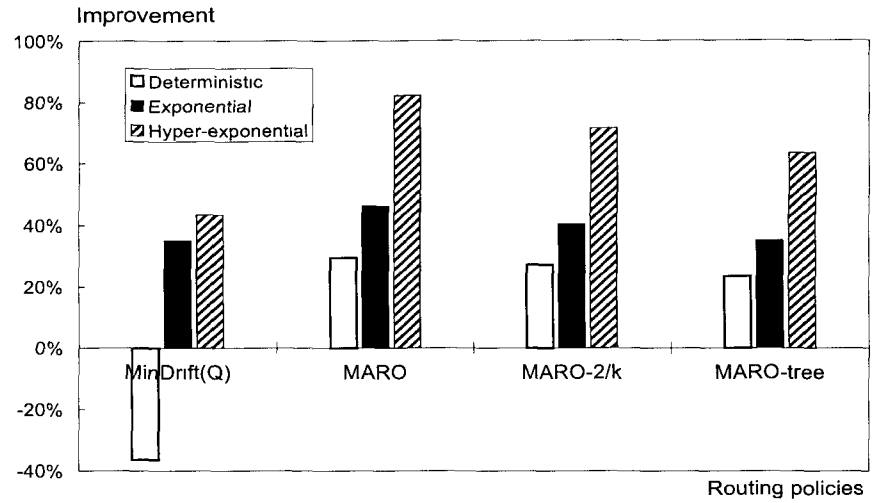

Figure 5.10: Processing time variance vs. Improvement of total mean queue length (System D, $\rho^* = 0.41$)

5.3 Hospital Waiting Times

Throughout the health care sector in Canada, waiting has been and continues to be the number one barrier for those having difficulties accessing medical services. Reports from Statistics Canada [10] indicate that nationwide median waiting times for all specialized services (e.g., specialist visits, diagnostic tests like non-emergency MRI/CT scans) have been at 3 to 4 weeks in 2005 and most individuals continue to report that they received the services within 3 months, depending on the kind of care. In some densely populated places, for example the Greater Toronto Area, the waiting times for CT scans are at 7 to 12 weeks from December 2005 to July 2006. Correspondingly, the CT-machine-to-population ratio is approximately $1/100,000$ and the number of CT scans per 1,000 population is about 80 in the 2005/2006 fiscal year [55]. Such long waiting times and large demands imply that the medical facilities have been operating with heavy loads.

As an important part of "Ontario's Wait Time Strategy" [38], the Ministry of Health and Long-Term Care has launched a single wait time information system that collects data from about 50 Ontario hospitals. Patients can also access the reports from this system online (http://www.health.gov.on.ca/

transformation/wait_times/public/wt_more_info.html#, accessed on November 30, 2007) to find out the waiting times over the past 2 months for specialized services in the hospitals around their neighbourhood (e.g., within 50 km), so that they are able to make informed choices about where to be referred for quicker service.

The insights obtained from the Grid system should also apply to the hospital system, since both can be modelled by the output-queued model in heavy traffic. We will demonstrate two resource allocation rules which are related to the above Wait Time Strategy. The service demands considered are mainly from outpatients, because it is the outpatients rather than the inpatients who need the waiting time information to decide which hospital to go to for quicker service.

5.3.1 The Model

Consider an area with J hospitals, each having a fixed capacity C_j for outpatient exams (e.g., CT scans). The capacity is the number of 15 minute (CT scan) slots that can be scheduled for any given day. In reality, as the data collected in [56] for Vancouver General Hospital indicate, exam lengths vary (in this case from 15 to 60 minutes), each being a multiplier of 15. Let type i exams have the length of $(15 \times i)$ minutes, $i = 1, ..., 4$. From a booking point of view, it is the scheduled length of the exam rather than the actual length that is relevant. Therefore, the service time distribution for type i exams at hospital j is assumed deterministic with mean $\mu_{i,j}^{-1} = (C_j/i)^{-1}$, i.e., the maximum number of type i exams can be scheduled at hospital j is $\mu_{i,j}$ per day.

It is known from the Grid system application (Figure 5.9) that for systems operating in heavy traffic, a small proportion of flexible arrivals can yield significant improvement in system performance, because a fraction of incoming workload can be shifted in the system, which leads to complete resource pooling. Therefore, given $0 \leq q \leq 1$, we assume that there are two pools of outpatients: $100(1-q)$ percent are the dedicated outpatients who always go to a particular hospital (e.g., the one in a patient's neighbourhood); $100q$ percent are the flexible ones who are willing to go to any one of the J

hospitals.

Assume that (1) the total outpatient demand (or the number of exams per day) is a Poisson random variable with mean $\tilde{\lambda}$; (2) the dedicated outpatient demand at each hospital j is proportional to the hospital capacity C_j. Then the flexible demand and each dedicated demand j are independent Poisson random variables with mean $\lambda^f = q\tilde{\lambda}$ and $\lambda_j^d = (1-q)\tilde{\lambda}C_j/\sum_{j=1}^{J} C_j$, respectively. Let p_i denote the probability of type i exams, $i = 1, ..., 4$. Then the total number of type i exams per day is also a Poisson random variable with mean $\lambda_i = \tilde{\lambda}p_i$.

Let $X_{i,j}^m$ denote the actual number of type i exams that can be scheduled at hospital j on day m. We have $\sum_{i=1}^{4} iX_{i,j}^m \leq C_j$. Let $Q_{i,j}^m$ denote the actual outpatient demand, which consists of the flexible part $F_{i,j}^m$ and the dedicated part $D_{i,j}^m$. Then the number of exams left over is

$$L_{i,j}^m = \max\left(0,\ (Q_{i,j}^m - X_{i,j}^m)\right).$$

If $L_{i,j}^m > 0$, it is included in the actual demand of the next day $Q_{i,j}^{m+1}$.

An effective resource allocation rule should yield smaller total leftover exams for the J hospitals. We compare two allocation rules which direct a flexible outpatient to a hospital in different ways.

Policy 5.3.1 (Decentralized rule). *Flexible outpatients make the decision on their own. Suppose the system provides daily values of the expected waiting time for type i exams at each hospital j, based on the number of left-over exams on day m and the dedicated demand known for the coming day $m+1$. Therefore, given the time*

$$W_{i,j} = \left\{ \sum_{n=1}^{4} \frac{D_{n,j}^{m+1} + L_{n,j}^m}{\mu_{n,j}} + \frac{1}{\mu_{i,j}} \right\}, \tag{5.9}$$

a flexible outpatient who needs a type i exam will choose hospital j which has the minimal $W_{i,j}$ on day $m+1$.

Policy 5.3.2 (Centralized rule). *Flexible outpatients stand by as "on call" for the next day exams. The system will refer a flexible outpatient who needs a*

type i exam to hospital j satisfying

$$j \in \arg\min_{j\in\mathcal{J}} \left\{ \sum_{n=1}^{4} \frac{Q_{n,j}^{m+1}(t)}{\mu_{n,j}} + \frac{1}{\mu_{i,j}} \right\}$$

for day $m+1$, where $Q_{n,j}^{m+1}(t)$ is the total number of type n exams demand at the time before notifying the flexible outpatient.

The decentralized rule is close to what has been described in "Ontario's Wait Time Strategy", except that the state information provided is updated more frequently and contains some future expectations. The centralized rule is the same as the minimum completion time policy [46], which can be seen as a special case of MARO with the cost function being $C(Z_j) = 0.5\mu_{i,j}Z_j^2 + Z_j$ for all $j \in \mathcal{J}$ and $Z_j = \sum_{n=1}^{4} \mu_{n,j}^{-1} Q_{n,j}(t)$. The difference between the two rules is that using the decentralized rule, all flexible outpatients who need the same type of exams will go to the same hospital which has the shortest expected waiting time, while using the centralized rule, the flexible demand can be shifted between hospitals in the short term in order to balance the loads.

5.3.2 Results

To compare the above two rules, we use simulations to examine the total leftover exams of all J hospitals over a period of $M/2$ days (with the first half of the M days set aside as a warm up period). The performance measure is the average total of leftover exams

$$\tilde{L} = \frac{2}{M} \sum_{m=\frac{M}{2}}^{M} L^m = \frac{2}{M} \sum_{m=\frac{M}{2}}^{M} \sum_{j=1}^{J} \sum_{i=1}^{I} L_{i,j}^m,$$

if the system is stable; or otherwise the growth rate of leftover exams

$$R = \frac{L^M - L^{\frac{M}{2}}}{M/2}.$$

Two trial systems are used:

System $\mathbf{H_1}$ has $J = 2$ identical hospitals, with capacities $C_1 = C_2 = 50$. So the mean service rate matrix is

$$\mu = \begin{bmatrix} 50.0 & 50.0 \\ 25.0 & 25.0 \\ 16.7 & 16.7 \\ 12.5 & 12.5 \end{bmatrix}.$$

The probability of type i exams is given by the i-th element of the vector

$$p^t = \begin{bmatrix} 0.5 & 0.3 & 0.1 & 0.1 \end{bmatrix}.$$

The total exam demand per day has mean $\tilde{\lambda} = 52.7$, so that the solution to LP (3.2) is $\rho^* = 0.95$.

System $\mathbf{H_2}$ has $J = 3$ different hospitals, with capacities $C_1 = 50$, $C_2 = 30$ and $C_3 = 20$, respectively, so the total capacity is the same as that of System $\mathbf{H_1}$. The mean service rate matrix is

$$\mu = \begin{bmatrix} 50.0 & 30.0 & 20.0 \\ 25.0 & 15.0 & 10.0 \\ 16.7 & 10.0 & 6.7 \\ 12.5 & 7.5 & 5.0 \end{bmatrix}.$$

The total exam demand per day has the same distribution as that of System $\mathbf{H_1}$ (i.e., the same $\tilde{\lambda}$ and p^t), and the solution ρ^* to LP (3.2) is also the same.

The simulation consists of 5000 replications of $M = 1000$ days each for a system. The results are at 95 percent confidence level, with accuracies no worse than 5%.

Table 5.18 compares the number of leftover exams at different flexibility levels q (assuming q is the same for all exam types), when the two allocation rules are applied to System H_1. $Imprv_1$ is the relative improvement over zero flexibility and $Imprv_2$ is the improvement of the centralized rule over the decentralized rule. It can be seen that System H_1 remains stable when the proportion of flexible demand increases. With a small amount of flexibility q, the centralized rule can achieve similar improvement as that achieved with large q. However, when the decentralized rule is applied, the proportion of flexible demand must be kept small, otherwise the performance gets worse.

This can be explained by the "herding effect" [50], where large amounts of flexible demand all choose to go to one hospital and result in congestion. This is avoided to some extent by the centralized rule.

In the decentralized rule, the state information on which the flexible outpatients base their (routing) decisions is updated on a daily basis. In the centralized rule, however, the system can obtain the updated state information instantly after each flexible demand is assigned. The simulation results suggest that an effective resource allocation policy should be able to obtain the state information that is updated at a speed comparable to the decision making frequency.

Table 5.19 compares the number and the growth rate of leftover exams in System H_2, where one of the hospitals in System H_1 is split into two hospitals with smaller capacities. It can be seen that when there is a large amount of flexible demand, both allocation rules can yield an unstable system. It has been observed that when the flexibility level q increases, the total mean utilized capacity $\hat{C}$ decreases as a result of more flexible demand going to the large hospital, leaving the small hospital under utilized. For example, using the centralized rule, $\hat{C} = \sum_{j=1}^{3} \hat{C}_j$ drops from 84.8 (out of 100) to 80.8 when q increases from 0.5 to 0.9, where the mean utilized capacity of the small hospital $\hat{C}_3$ decreases from 8.5 (out of 20) to 1.7, at the same time $\hat{C}_1$ (of the large hospital) increases from 47.8 (out of 50) to 49.5.

Although the centralized rule continues to outperform the decentralized rule when the system is stable, both rules appear the same when the system becomes unstable. This is not surprising, since the system load is unchanged no matter which allocation rule is applied, so the rate of divergence of the total demand per day Q^m is the same for both rules. Given $Q^m = L^m + \tilde{\lambda}$ (the amount of leftover exams plus the fixed throughput), L^m diverges at the same speed as Q^m does, when the system is unstable.

The simulation results suggest that when using a policy that has optimality properties in the heavy traffic limit, the system load should approach the critical value from below, rather than from above, so that the system remains stable at any time. Otherwise, the "optimal" policy appears the same as any policy that keeps all servers busy.

Table 5.18: Leftover exams in System H_1

	Centralized Rule		**Decentralized Rule**		
q	$\tilde{L}$	$\mathbf{Imprv_1}$	$\tilde{L}$	$\mathbf{Imprv_1}$	$\mathbf{Imprv_2}$
0	2.62	–	2.62	–	0%
0.1	1.17	56%	1.38	47%	16%
0.3	0.80	70%	3.67	−40%	78%
0.5	0.76	71%	9.03	−245%	92%
0.7	0.73	72%	14.99	−472%	95%
0.9	0.72	73%	20.70	−690%	97%

Table 5.19: Leftover exams in System H_2

	Centralized Rule		**Decentralized Rule**		
q	$\tilde{L}$	$\mathbf{Imprv_1}$	$\tilde{L}$	$\mathbf{Imprv_1}$	$\mathbf{Imprv_2}$
0	5.45	–	5.45	–	0%
0.1	2.79	49%	3.23	41%	13%
0.2	2.71	50%	4.94	9%	45%
0.3	3.91	28%	9.06	−66%	57%
0.4	6.23	−14%	15.60	−186%	60%
0.5	12.58	−131%	26.70	−390%	53%
q	R		R		$\mathbf{Imprv_2}$
0.7	0.40		0.40		0%
0.9	2.53		2.53		0%

5.4 Summary

Simulation studies of the MARO related policies have the following implications, which provide guidelines in designing the routing policies of an output-queued system.

In the case of a single arrival type, we have that

- The performance improvement achieved by the MARO related policies is close to the lower bound of achievable performance, when the system load is high.

- Such improvement is larger in systems with larger service time variance, because the proposed dynamic routing policies allow incoming workload to be freely shifted over short time scales between all the queues in the system.

- Using significantly less state information, MARO$-2/k$ and MARO$-$tree achieve improvement competitive with the dynamic policy which requires global state information, although there is a tradeoff between the system performance and the cost of acquiring state information for making routing decisions.

- A small number of flexible arrivals can yield significant performance improvement, especially when the system is operating with high loads. If the flexibility of choosing servers is limited (e.g., due to locality constraints), the ring routing structure is a good choice; otherwise, MARO$-2/k$ is recommended.

In the case of multiple arrival types, we have that

- The MARO related policies outperform the MinDrift(Q) policy in heterogeneous server systems with either high or medium loads, while requiring significantly less state information. This is achieved by routing arrivals to the "appropriate" subsets of the queues, with the aid of a resource allocation LP. The routing structure does not change if the mean arrival rates of all task types vary by the same factor.

- For homogeneous server systems, MinDrift$-2/k$ is recommended to spread the incoming workload over all of the queues and at the same time to reduce the amount of state information required for routing.
- The improvement over the static routing policy is larger in systems with a larger number of servers, or with higher loads, or with larger service time variance.
- When the system size grows, MARO$-2/k$ and MARO$-$tree yield larger discounts of required state information than MARO$-$flex. However, the relative values of the system performance and the cost of acquiring state information should be evaluated, before choosing one of the MARO related policies with an appropriate flexibility level.
- An effective dynamic routing policy should be able to obtain state information that is updated at a speed comparable to the decision making frequency.

Chapter 6

Conclusions and Future Work

In this thesis, we have studied several resource allocation policies which require varying amounts of information for output-queued systems with multiple types of tasks. These policies require no knowledge of the actual processing times of the tasks for making routing decisions, but use the first (and in some cases the second) moments of the task inter-arrival times and processing times, with a small amount of state information (or even none), in order to minimize the delay in the system.

For systems with heterogeneous servers, we proposed the MinDrift Affinity Routing (MARO) policy and three variants, namely MARO$-2/k$, MARO$-$flex and MARO$-$tree. These policies are designed to maximize the capacity of the system by using the first moments of the task inter-arrival times and processing times, and to minimize the delay in the system by using a limited amount of state information. Specifically, the state information is the expected increment of the aggregate (convex) holding cost of multiple task types. As a special case, the state information can be the expected waiting time in systems with a single task type. On average, the amount of state information needed is less than half of the global state information. This amount can be further reduced if the flexibility level is low, e.g., only a certain proportion of time is available for the dispatcher (who makes the routing decisions) to acquire state information, or only a certain proportion of tasks are flexible, or the flexible tasks can only afford a small number of dynamic choices. Nonetheless, using diffusion limits for systems with Poisson arrival processes,

we prove that MARO, MARO−flex and MARO−tree have the same heavy traffic optimality properties and the optimality is achieved independent of the levels of flexibility. For systems with a single task type, the MARO related policies approach the lower bound of achievable performance (where no routing is required).

By demonstrating applications of the MARO related policies in distributed computing systems and in medical services, we have shown that:

- It is important for a dynamic routing policy to allow a sufficient proportion of incoming workload to be shifted from one server to any other server in the system.
- A small number of flexible tasks can yield significant performance improvement (over the corresponding static policy), especially when the system is operating under high load.
- Such improvement is larger in systems with larger service time variance, or with a larger number of servers.
- Using the MARO related policies, each server in the system is allowed to be capable of processing a small number of different task types. This is desirable when it is costly to maintain highly flexible servers.
- By routing the tasks to the "appropriate" subsets of the servers, the MARO related policies (which require significantly less state information) can outperform the MinDrift(Q) policy (which requires global state information), in heterogeneous server systems with either high loads or medium loads.
- When the system size grows, the MARO related policies yield larger discounts of required state information. However, the relative values of the system performance and the cost of acquiring state information should be evaluated, before choosing one of the policies with a proper flexibility level.
- An effective dynamic routing policy should be able to obtain updated state information at a rate comparable to the decision making frequency.

We have also proposed resource allocation policies which require no state information. For output-queued systems which are operating under heavy traffic, the random routing policy asymptotically minimizes the delay in the system by using the second moments of the task processing times. The accompanying pooling strategy further reduces the delay by combining appropriate parallel single-server queues into a number of multi-server queues.

The proposed random routing policy has several features. First, it only requires each server in the system to be capable of processing a small number of different task types, in fact a number of servers are dedicated to processing one type of tasks. This is desirable when the cost is high to maintain highly flexible servers. Second, it routes a number of the same task types to a subset of identical servers, which enables these servers to be pooled together, so that the system performance can be further improved. Third, it is optimal for various processing time distributions whose (squared) coefficients of variation are differentiated by a constant factor, if the arrival streams of tasks follow Poisson processes.

We have shown that to minimize the delay in the system, the pooling strategy depends on the combinations of the first two moments of the task inter-arrival times and processing times. There are cases where (partial) pooling subsets of the servers is better than (full) pooling all of the identical servers into a single queue. In cases where full pooling is optimal, there is still a tradeoff between the system performance and the cost of maintaining highly flexible servers, since pooling requires the corresponding servers to be capable of processing the same types of tasks.

There are several lines along which future research could proceed.

- It is noted that with an arbitrary work-conserving local scheduling rule, MARO (as well as its variants) asymptotically minimizes the delay in either the input-queued or output-queued system. For an input-queued system, MARO can still be used to determine which servers to route to for tasks waiting in their dedicated queues. Therefore, a study of MARO's application in staffing service operations (e.g., call centres) would be of interest.

- The MARO related policies are known to be robust in the sense that the routing structure does not change if the mean arrival rates of all the task types vary by the same factor. It will be of interest to study how the estimates of the mean arrival rates affect the routing structure, if the estimates are apart from the true values by different factors for multiple task types.

- As for the local scheduling policy, MARO allows preemption of service or server sharing for tasks of different types. Anantharam showed in [3] that preemptive policies are better than non-preemptive policies in the sense of decreasing mean waiting time under high variance. How to compare the performance between an output-queued system using MARO and a fully-pooled queue with the same heterogeneous servers, both using the same preemptive local scheduling policy, would be of interest to study.

- To obtain the optimal routing policy which requires no state information, we have tried several means of generating the random routing matrix, instead of using a generic nonlinear programming solver. Fine-tuning the heuristic procedures so that the solutions are closer to the optimal values given by the NLP solvers might be useful. In particularly, when the system parameters change frequently so that the optimal routing policy needs to be recalculated, using the heuristics will significantly reduce the time spent in making routing decisions.

Bibliography

[1] A. O. Allen. *Probability, Statistics and Queueing Theory with Computer Science Applications*, Academic Press, second edition, 1990.

[2] S. Altschul, W. Gish, W. Miller, E. Myers and D. Lipman. A Basic Local Alignment Search Tool. *Journal of Molecular Biology*, **215**:403–410, 1990.

[3] V. Anantharam. Scheduling Strategies and Long-range Dependence. *Queueing Systmes*, **33**(1-3):73–89, 1999.

[4] M. Armony. Dynamic Routing in Large-Scale Service Systems with Heterogeneous Servers. *Queueing Systems*, **51**:287–329, 2005.

[5] M. Armony and C. Maglaras. On Customer Contact Centers with a Call-Back Option: Customer Decisions, Routing Rules and System Design. *Operations Research*, **52**(2):271–292, 2004.

[6] R. Atar, A. Mandelbaum and M. Reiman. Scheduling a Multi-class Queue with Many Exponential Servers: Asymptotic Optimality in Heavy Traffic. *Annals of Applied Probability*, **14**(3):1084–1134, 2004.

[7] S. Andradóttir, H. Ayhan and D. G. Down. Dynamic Server Allocation for Queueing Networks with Flexible Servers. *Operations Research*, **51**(6):952–968, 2003.

[8] A. Bassamboo, J. M. Harrison and A. Zeevi. Design and Control of a Large Call Center: Asymptotic Analysis of an LP-based Method. *Operations Research*, **54**(3):419–435, 2006.

[9] C. Beightler and D. Phillips. *Applied Geometric Programming.* Wiley, New York, 1976.

[10] J. M. Berthelot and C. Sanmartin. *Access to Health Care Services in Canada.* Statistics Canada, Catalogue 82-575-XWE, 2006.

[11] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*, Athena Scientific, 1997.

[12] P. Billingsley. *Convergence of Probability Measures*, John Wiley and Sons, 1968.

[13] K. Birman. Can Web Services Scale Up? *IEEE Computer*, **38**(10):107–110, 2005.

[14] S. Boyd and L. Vandenberghe. *Convex Optimization*, Cambridge University Press, Cambridge, 2004.

[15] J. A. Buzacott. Commonalities in Re-engineered Business Processes: Models and Issues. *Management Science*, **42**:768–782, 1996.

[16] J. A. Buzacott and J. G. Shanthikumar. *Stochastic Models of Manufacturing Systems*, Prentice Hall, 1993.

[17] J. W. Byers, J. Considine and M. Mitzenmacher. Geometric Generalizations of the Power of Two Choices. *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, Barcelona, 54–63, 2004.

[18] R. H. Byrd, J. Nocedal and R. A. Waltz. KNITRO: An Integrated Package for Nonlinear Optimization. G. Di Pillo and M. Roma (eds.), *Large-Scale Nonlinear Optimization*, **83**:35–59, Springer, Netherlands, 2006.

[19] H. Chen and D. D. Yao. *Fundamentals of Queueing Networks: Performance, Asymptotics and Optimization*, Springer, New York, 2001.

[20] E. D. Dolan, J. J. Moré and T. S. Munson. Benchmarking Optimization Software with COPS 3.0, Technical Report ANL/MCS-273, Argonne National Laboratory, 2004.

[21] H. Feng, V. Misra, D. Rubenstein. Optimal State-free, Size-Aware Dispatching for Heterogenous $M/G/-$type Systems. *Performance Evaluation*, **62**:475–492, 2005.

[22] R. D. Foley and D. R. McDonald. Join the Shortest Queue: Stability and Exact Asymptotics. *The Annals of Applied Probability*, **11**(3):569–607, 2001.

[23] I. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, **15**(3):200–222, 2001.

[24] R. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel. Scheduling Resources in Multi-User, Heterogeneous Computing Environments with SmartNet. *Proceedings of the 7th Heterogeneous ComputingWorkshop*, 184–199, 1998.

[25] N. Gans, G. Koole and A. Mandelbaum. Telephone Call Centers: Tutorial, Review, and Research Prospects. *Manufacturing and Service Operations Management*, **5**(2):79–141, 2003.

[26] O. Garnett and A. Mandelbaum. An Introduction to Skills-based Routing and Its Operational Complexities. Teaching Notes, Technion, Haifa, Israel, 2001.

[27] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Programming. Technical Report SOL 97-3, Systems Optimization Laboratory, Stanford University, 1997.

[28] C. Graham. Functional central limit theorems for a large network in which customers join the shortest of several queues. *Probab. Theory Relat. Fields*, **131**:97–120, 2005.

[29] D. Gross and C. Harris. *Fundamentals of Queueing Theory*, Wiley-Interscience, third edition, 1998.

[30] S. Gurumurthi and S. Benjaafar. Modeling and Analysis of Flexible Queueing Systems. *Naval Research Logistics*, **51**:755–782, 2004.

[31] S. Halfin and W. Whitt. Heavy Traffic Limits for Queues with Many Exponential Servers. *Operations Research*, **29**:567–587, 1981.

[32] R. W. Hall. *Queueing Methods for Services and Manufacturing*, Prentice Hall, 1991.

[33] P. Hansen, B. Jaumard and S.H. Lu. Some Further Results on Monotonicity in Globally Optimal Design. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, **111**(3):345–352, 1989.

[34] J. M. Harrison. *Brownian Motion and Stochastic Flow System*, Wiley, New York, 1985.

[35] J. M. Harrison and M. J. López. Heavy Traffic Resource Pooling in Parallel-server Systems. *Queueing Systems*, **33**:339–368, 1999.

[36] Y.-T. He, I. Al-azzoni and D. G. Down. MARO – MinDrift Affinity Routing for Resource Management in Heterogeneous Computing Systems. B. Spencer, M.-A. Storey and D. Stewart (eds.), *Proceedings of the 17th Annual Conference of IBM Centre for Advanced Studies on Collaborative Research* (CASCON'07), 71–85, 2007.

[37] K. Holmström, A. O. Göran and M. M. Edvall. *User's Guide for TOMLAB/KNITRO v5.01*, TOMLAB Optimization Inc., 2006.

[38] A. R. Hudson and P. Glynn *Ontario's Wait Time Strategy: Overview*, Ontario Ministry of Health and Long-Term Care, December, 2004. Available via: http://www.health.gov.on.ca/transformation/wait_times/providers/wt_strategy.html

[39] D. L. Iglehart and W. Whitt. Multiple Channel Queues in Heavy Traffic, I and II. *Advances in Applied Probability*, **2**:150–177, 355–364, 1970.

[40] H. Jia and F. Ordónez and M. M. Dessouky. Solution Approaches for Facility Location of Medical Supplies for Large-Scale Emergencies. *Computers and Industrial Engineering.* **52**(2):257–276, 2007.

[41] J. F. C. Kingman. The Heavy Traffic Approximation in the Theory of Queues. *Proceedings of the Symposium on Congestion Theory*, W. Smith and W. Wilkinson (eds.), University of North Carolina Press, 137–159, 1965.

[42] L. Kleinrock. *Queueing System, Volume II: Computer Applications*, Wiley, New York, 1976.

[43] L. Kontothanassis and D. Goddeau. Profile Driven Scheduling for a Heterogeneous Server Cluster. *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW'05)*, 336–345, IEEE Computer Society, 2005.

[44] K. Krauter R. Buyya and M. Maheswaran. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. *Software Practice Experience*, **32**(2):135–164, 2002.

[45] D. Lu, H. Y. Sheng, and P. A. Dinda. Effects and Implications of File Size/Service Time Correlation on Web Server Scheduling Policy. *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (MASCOTS'05), 2005.

[46] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen and R. F. Freund. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems. *Proceedings of the 8th Heterogeneous Computing Workshop*, 30–44, 1999.

[47] A. Mandelbaum and M. Reiman. On Pooling in Queueing Networks. *Management Science*, **44**(7):971–981, 1998.

[48] A. Mandelbaum and A. L. Stolyar. Scheduling Flexible Servers with Convex Delay Costs: Heavy-Traffic Optimality of the Generalized $c\mu$–Rule. *Operations Research*, **52**(6):836–855, 2004.

[49] C. Maranas and C. Floudas. Global Optimization in Generalized Geometric Programming. *Computers and Chemical Engineering*, **21**(4):351–370, 1997.

[50] M. Mitzenmacher. How Useful Is Old Information? *IEEE Transactions on Parallel Distributed Systems*, **11**(1):6–20, 2000.

[51] M. Mitzenmacher. The Power of Two Choices in Randomized Load Balancing. *IEEE Transactions on Parallel and Distributed Systems*, **12**(10):1094–1104, 2001.

[52] J. L. Morales, J. Nocedal, R. Waltz, G. Liu, and J. P. Goux. Assessing the Potential of Interior Methods for Nonlinear Optimization. O. Ghattas (ed.), *Proceedings of the First Sandia Workshop on Large-Scale PDE-Constrained Optimization*, Springer Verlag, 2002.

[53] *The MOSEK optimization toolbox for MATLAB manual. Version 4.0*, MOSEK ApS, Denmark, 2006.

[54] B. A. Murtagh and M. A. Saunders. MINOS 5.5 User's Guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Stanford University, 1998.

[55] Ontario Ministry of Health and Long-Term Care. *Ontario's Wait Time Strategy: MRI & CT Expert Panel - Phase II Report*, December, 2006. Available via: http://www.health.gov.on.ca/transformation/wait_times/providers/wt_strategy.html

[56] J. Patrick and M. L. Puterman. Improving Resource Utilization for Diagnostic Services through Flexible Inpatient Scheduling: A Method for Improving Resource Utilization. *Journal of the Operational Research Society*, **58**:235–245, 2007.

[57] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, New York, 1994.

[58] M. Reiman. Some Diffusion Approximations with State Space Collapse. *Modelling and Performance Evaluation Methodology*, F. Baccelli and G. Fayolle (eds.), Lecture Notes in Control and Information Sciences, **60**:209–240, Springer, 1984.

[59] A. Sharifnia Instability of the Join-The-Shortest-Queue and FCFS Policies in Queuing Systems and Their Stabilization. *Operations Research*, **45**(2):309–314, 1997.

[60] R. A. Shumsky. Approximation and Analysis of a Call Center with Flexible and Specialized Servers. *OR Spectrum, Special Issue on Call Centre Management*, **26**(3):307–330, 2004.

[61] D. R. Smith and W. Whitt. Resource Sharing for Efficiency in Traffic Systems. *Bell System Technical Journal*, **60**:39–55, 1981.

[62] A. L. Stolyar. MaxWeight Scheduling in a Generalized Switch: State Space Collapse and Workload Minimization in Heavy Traffic. *Annals of Applied Probability*, **14**(1): 1–53, 2004.

[63] A. L. Stolyar. Optimal Routing in Output-Queued Flexible Server Systems. *Probability in the Engineering and Informational Science*, **19**(2):141–189, 2005.

[64] Y.-C. Teh and A. R. Ward. Critical Thresholds for Dynamic Routing in Queueing Networks. *Queueing Systems*, **42**:297–316, 2002.

[65] E. Tekin, W. J. Hopp and M. P. Van Oyen. Pooling Strategies for Call Centre Agent Cross-training. Working paper, Northwestern University, 2004.

[66] W. Whitt. Deciding Which Queue to Join: Some Counter Examples. *Operations Research*, **34**:226–244, 1986.

[67] W. Whitt. Approximations for the $GI/G/m$ Queue. *Production and Operations Management*, **2**:114–161, 1993.

[68] W. Whitt. *Stochastic-Processes Limits: An Introduction to Stochastic-Process Limits and Their Application to Queues*, Springer, New York, 2002.

[69] R. J. Williams. On Dynamic Scheduling of a Parallel Server System with Complete Resource Pooling in Heavy Traffic. *Analysis of Communication Networks: Call Centres, Traffic and Performance*, D. McDonald and S.R.E. Turner (eds.), Fields Institute Communication Series, **28**:49–71, American Mathematical Society, 2000.

[70] W. Winston. Optimality of the Shortest Line Discipline. *Journal of Applied Probability*, **14**:181–189, 1977.

[71] R. Wu. *Scalable Scheduling of Parallel Servers*, Ph.D. thesis, McMaster University, 2007.

[72] H. Zhang and G.-H. Hsu. Heavy Traffic Limit Theorems for a Sequence of Shortest Queueing Systems. *Queueing Systems*, **21**:217–238, 1995.

Appendix A

An Experiment of Solving the Resource Allocation NLP Using Geometric Programming

In this section, we discuss an alternative way of solving the NLP (4.24) using geometric programming (GP). Although our experiment on a sample problem was not successful with MOSEK's GP solver [53], we present here the method of transforming the original problem into a series of standard form GP problems, while each standard GP can be converted to a convex optimization problem using logarithmic transformation [14]. What remains an open question is whether our attempt can be used as a starting point for exploiting the well-developed convex optimization techniques for the NLP problems which are like the ones described in this thesis.

A.1 Standard GP

The standard GP is of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & f_m(\mathbf{x}) \leq 1, \quad 1 \leq m \leq M \\ & h_\ell(\mathbf{x}) = 1, \quad 1 \leq \ell \leq L. \end{aligned} \tag{A.1}$$

The decision variables $\mathbf{x} \in \mathbb{R}^N_{++}$ are strictly positive. The objective function f_0 and the inequality constraints f_m are posynomials of the form

$$f_m(\mathbf{x}) = \sum_{k=1}^{K_m} c_{m,k} \prod_{i=1}^{N} x_i^{a_{m,k,i}}, \quad 0 \le m \le M;$$

the equality constraints h_ℓ are monomials of the form

$$h_\ell(\mathbf{x}) = c_\ell \prod_{i=1}^{N} x_i^{a_{\ell,i}}, \quad 0 \le \ell \le L,$$

where all coefficients $c_{m,k}$ and c_ℓ are strictly positive; all exponents $a_{m,k,i}$ and $a_{\ell,i}$ are real-valued. (If there exists a negative coefficient $c_{m,k}$, f_m is called signomial.)

GP (A.1) can be converted to a convex problem using a logarithmic transformation of the variables, the objective and the constraint functions. Let

$$\begin{aligned}
\tilde{\mathbf{x}} &= [\log x_1, \ldots, \log x_N]^T, \\
\tilde{c}_{m,k} &= \log c_{m,k}, \quad 0 \le m \le M, \quad 1 \le k \le K_m \\
\mathbf{a}_{m,k} &= [a_{m,k,1}, \ldots, a_{m,k,N}], \quad 0 \le m \le M, \quad 1 \le k \le K_m \\
\mathbf{a}_\ell &= [a_{\ell,1}, \ldots, a_{\ell,N}], \quad 0 \le \ell \le L.
\end{aligned}$$

GP (A.1) in convex form is

$$\begin{aligned}
\min_{\tilde{\mathbf{x}}} \quad & \tilde{f}_0(\tilde{\mathbf{x}}) = \log \left(\sum_{k=1}^{K_0} e^{\mathbf{a}_{0,k} \cdot \tilde{\mathbf{x}} + \tilde{c}_{0,k}} \right) \\
\text{s.t.} \quad & \tilde{f}_m(\tilde{\mathbf{x}}) = \log \left(\sum_{k=1}^{K_m} e^{\mathbf{a}_{m,k}\, \tilde{\mathbf{x}} + \tilde{c}_{m,k}} \right) \le 0, \quad 1 \le m \le M \\
& \tilde{h}_\ell(\tilde{\mathbf{x}}) = \mathbf{a}_\ell \cdot \tilde{\mathbf{x}} + \log c_\ell = 0, \quad 1 \le \ell \le L.
\end{aligned} \tag{A.2}$$

A high quality implementation of a GP solver is available in the MOSEK software package. MOSEK takes as inputs the coefficients $c_{m,k}$, c_ℓ and exponents $a_{m,k,i}$, $a_{\ell,i}$ and obtains GP (A.2) as a result of pre-processing. Then the convex form is solved using a primal-dual interior-point method [14].

A.2 Reformulation of the NLP Problem

Let $\lambda \in \mathbb{R}_+^I$, $\rho \in \mathbb{R}_+^I$ and $\theta \in \mathbb{R}_+^I$ be vectors of positive parameters. We denote by $\mathbf{y} \in \mathbb{R}_+^{IJ+3J}$ the vector of decision variables. The original NLP is

$$\min_{\mathbf{y}} \quad f_0(\mathbf{y}) = \sum_{j=IJ+1}^{IJ+J} \left(y_j^2\right)\left(y_{j+J}\right)\left(y_{j+2J}^{-1}\right) \tag{A.3}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \lambda_i y_{(j-1)I+i} = y_{j+IJ}, \quad 1 \le j \le J \tag{A.4}$$

$$\sum_{i=1}^{I} \theta_i y_{(j-1)I+i} = y_{j+IJ+J}, \quad 1 \le j \le J \tag{A.5}$$

$$\sum_{i=1}^{I} y_{(j-1)I+i} = y_{j+IJ+2J}, \quad 1 \le j \le J \tag{A.6}$$

$$\sum_{i=1}^{I} \rho_i y_{(j-1)I+i} = 1, \quad 1 \le j \le J \tag{A.7}$$

$$\sum_{j=1}^{J} y_{(j-1)I+i} = 1, \quad 1 \le i \le I \tag{A.8}$$

$$0 \le y_{(j-1)I+i} \le 1, \quad 1 \le i \le I, \quad 1 \le j \le J \tag{A.9}$$

$$0 \le y_{j+IJ} \le \sum_{i=1}^{I} \lambda_i, \quad 0 \le y_{j+IJ+J} \le \sum_{i=1}^{I} \theta_i,$$
$$0 \le y_{j+IJ+2J} \le I, \quad 1 \le j \le J. \tag{A.10}$$

Constraint (A.9) means that the first IJ variables $y_{(j-1)I+i}$ correspond to the routing probabilities $p_{i,j}$. Linear combinations of these IJ variables yield the remaining $3J$ variables, as given by constraints (A.4)–(A.6).

To transform the original NLP into a series of standard GP problems, we first make the assumption that all of the decision variables in (A.3) are strictly positive. Let $\mathbf{x_1} \in \mathbb{R}_{++}^{IJ}$ denote the vector of new decision variables, where $x_i = y_i + \epsilon$ with small constant $\epsilon > 0$, for all $1 \le i \le (IJ)$. If the solution $x_i^* < 2\epsilon$, the optimal value of y_i^* is set to zero. For example, let $2\epsilon = 0.01$, any probability less than 0.005 is considered to be zero. Using (A.4)–(A.6), we construct the vector $\mathbf{x_2} \in \mathbb{R}_{++}^{3J}$ from $\mathbf{x_1}$.

To find a local optimal solution, (A.3) is reformulated as

$$\min_{(\mathbf{x_1},\mathbf{x_2})} \quad f_0(\mathbf{x_2}) = \sum_{j=IJ+1}^{IJ+J} \left(x_j^2\right)\left(x_{j+J}\right)\left(x_{j+2J}^{-1}\right) \tag{A.11}$$

$$\text{s.t.} \quad x_{j+IJ}^{-1} \sum_{i=1}^{I} \lambda_i x_{(j-1)I+i} \leq 1, \quad 1 \leq j \leq J \tag{A.12}$$

$$x_{j+IJ+J}^{-1} \sum_{i=1}^{I} \theta_i x_{(j-1)I+i} \leq 1, \quad 1 \leq j \leq J \tag{A.13}$$

$$x_{j+IJ+2J} \prod_{i=1}^{I} \gamma_{i,j}^{\gamma_{i,j}} \prod_{i=1}^{I} x_{(j-1)I+i}^{-\gamma_{i,j}} \leq 1, \quad 1 \leq j \leq J \tag{A.14}$$

$$\sum_{i=1}^{I} \rho_i x_{(j-1)I+i} \leq 1, \quad 1 \leq j \leq J \tag{A.15}$$

$$\prod_{i=1}^{I} \left(\frac{\alpha_{i,j}}{\rho_i}\right)^{\alpha_{i,j}} \prod_{i=1}^{I} x_{(j-1)I+i}^{-\alpha_{i,j}} \leq 1, \quad 1 \leq j \leq J \tag{A.16}$$

$$\sum_{j=1}^{J} x_{(j-1)I+i} \leq 1, \quad 1 \leq i \leq I \tag{A.17}$$

$$\prod_{j=1}^{J} \beta_{j,i}^{\beta_{j,i}} \prod_{j=1}^{J} x_{(j-1)I+i}^{-\beta_{j,i}} \leq 1, \quad 1 \leq i \leq I \tag{A.18}$$

$$x_i \leq 1, \quad 1 \leq i \leq IJ \tag{A.19}$$

$$\left(\sum_{i=1}^{I} \lambda_i\right)^{-1} x_{j+IJ} \leq 1, \quad 1 \leq j \leq J \tag{A.20}$$

$$\left(\sum_{i=1}^{I} \theta_i\right)^{-1} x_{j+IJ+J} \leq 1, \quad 1 \leq j \leq J \tag{A.21}$$

$$x_{j+IJ+2J}/I \leq 1, \quad 1 \leq j \leq J. \tag{A.22}$$

Constraints (A.12)–(A.14) are transformed from (A.4)–(A.6). For each x_{j+IJ} $(1 \leq j \leq J)$, the left-hand side of (A.12) is the only constraint function which is monotonically decreasing, while f_0 is monotonically increasing. Constraint (A.12) will be active at the minimum $(\mathbf{x}_1^*, \mathbf{x}_2^*)$ [33, 49]. Similarly, constraints (A.13)–(A.14) will be active.

Constraints (A.15) and (A.16), which are transformed from (A.7), provide upper and lower bounds to $x_{(j-1)I+i}$ $(1 \leq j \leq J)$, respectively. Specifi-

cally, (A.16) is derived by condensing the lower bound constraints (which are signomials)

$$\left(\rho_1 x_{(j-1)I+1}\right)^{-1} - \sum_{i=2}^{I} \left(\rho_1 x_{(j-1)I+1}\right)^{-1} \left(\rho_i x_{(j-1)I+i}\right) \leq 1, \quad 1 \leq j \leq J \text{ (A.23)}$$

using the geometric inequality [9]

$$\sum_k g_k(\mathbf{x}) \geq \prod_k \left(\frac{g_k(\mathbf{x})}{\alpha_k}\right)^{\alpha_k},$$

where $g_k(\mathbf{x})$ is monomial and $\sum_k \alpha_k = 1$, $(\forall k)\, \alpha_k > 0$. For all $1 \leq j \leq J$, let

$$g_{i,j}(\mathbf{x}_1) = \begin{cases} 1, & i = 1, \\ \left(\rho_1 x_{(j-1)I+1}\right)^{-1} \left(\rho_i x_{(j-1)I+i}\right), & 2 \leq i \leq I. \end{cases}$$

The parameter $\alpha_{i,j}$ is determined by

$$\alpha_{i,j} = g_{i,j}(\hat{\mathbf{x}}_1^{(0)}) / \sum_{i=1}^{I} g_{i,j}(\hat{\mathbf{x}}_1^{(0)}), \tag{A.24}$$

where $\hat{\mathbf{x}}_1^{(0)}$ is a feasible solution to the constraints (A.7)–(A.9). So we have $\sum_{i=1}^{I} \alpha_{i,j} = 1$, for all $1 \leq j \leq J$. Then (A.23) becomes

$$\left(\rho_1 x_{(j-1)I+1}\right)^{-1} \prod_{i=1}^{I} \left(\frac{g_{i,j}(\mathbf{x})}{\alpha_{i,j}}\right)^{-\alpha_{i,j}} \leq 1, \quad 1 \leq j \leq J,$$

which is the same as (A.16). Similarly, constraints (A.17) and (A.18) are transformed from (A.9).

Constraints (A.19)–(A.22) are transformed from (A.9)–(A.10).

To obtain the global optimum of the original NLP (A.3), we solve a series of GPs (A.11) using the following procedure.

- Step 1: Choose a feasible $\hat{\mathbf{x}}_1^{(0)}$ which satisfies (A.7)–(A.9). Let $t^{(0)} = f_0(\hat{\mathbf{x}}_2^{(0)})$, where $\hat{\mathbf{x}}_2^{(0)}$ is obtained using (A.4)–(A.6). Set $n = 1$.

- Step 2: Solve an augmented GP

$$\begin{aligned} \min_{(\mathbf{x}_1,\mathbf{x}_2,t^{(n)})} \quad & t^{(n)} \\ \text{s.t.} \quad & f_0(\mathbf{x}_2)\left(t^{(n)}\right)^{-1} \leq 1, \\ & \text{(A.12)} - \text{(A.22)}, \\ & \frac{s}{t^{(n-1)}} t^{(n)} \leq 1. \end{aligned} \tag{A.25}$$

  where s is a constant factor a bit larger than 1, e.g., $s = 1.01$.

- Step 3: If solving (A.25) is infeasible and $\hat{\mathbf{x}}_1^{(n)}$ is feasible, then the optimum is obtained and the procedure is terminated.

  If solving (A.25) is feasible and $\hat{\mathbf{x}}_1^{(n)}$ is also feasible, then (i) $t^{(n)} = f_0(\hat{\mathbf{x}}_2^{(n)})$; (ii) set $n = n + 1$ and go to Step 2.

  If solving (A.25) is infeasible and $\hat{\mathbf{x}}_1^{(n)}$ is also infeasible (i.e., any one of the constraints (A.12)–(A.22) is violated), then (i) use (A.24) to update the parameters $\alpha_{i,j}$ (and $\beta_{j,i}$) with $\hat{\mathbf{x}}_1^{(n)}$; (ii) $t^{(n)} = t^{(n-1)}$; (iii) set $n = n + 1$ and go to Step 2. (If $\hat{\mathbf{x}}_1^{(n)}$ is infeasible with a large n and $||\hat{\mathbf{x}}_1^{(n)} - \hat{\mathbf{x}}_1^{(n-1)}|| < \epsilon$, the procedure is terminated with an error.)

A.3 Numerical Examples

Let $I = J = 2$, $\lambda = [1, 0.3]$, $\rho = [1/2, 3/2]$, , $\theta = [2/4, 0.6/0.04]$. The global optimal solution to NLP (A.3) is known to be $\mathbf{x}_1^* = [1, 1/3, 0, 2/3]$.

The first experiment was trying to obtain a local optimum using GP (A.11). The starting point was $\mathbf{x}_1^{(0)} = [0.5, 0.5, 0.5, 0.5]$. The MOSEK GP solver returned the primal (and dual) feasible "optimal" solution $\mathbf{x}_1^* = \mathbf{x}_1^{(0)}$.

The second experiment was trying to obtain a global optimum using the augmented GP (A.25). The starting point was $\mathbf{x}_1^{(0)} = [0.5, 0.5, 0.5, 0.5]$. The MOSEK GP solver returned the primal (and dual) infeasible solution $\mathbf{x}_1 = [0.51, 0.51, 0.49, 0.53]$.

Both experiments did not yield correct answers. However, it is hoped that reformulating an NLP like (A.3) into a series of standard GPs can be applicable to other GP solvers.

Appendix B

A Resource Allocation Heuristic

Here we propose a heuristic to generate the routing probability matrix for an output-queued system which operates using static routing in heavy traffic. The system has J identical servers in parallel to process I types of tasks. The arrivals of type i tasks follow a Poisson process with rate λ_i and are routed to server j immediately upon arrival, with probability p_{ij}. The processing times of type i tasks have mean μ_i^{-1} and variance β_i^2. The local scheduling rule at each server is FCFS. The routing matrix obtained by the heuristic is trying to (1) decrease the total queue length of the system, where no state information is available and (2) increase the degree of pooling for the associated pooling strategy, which aims to further decrease the delay in the system.

In Section 4.1, we formulated the following NLP to obtain the optimal routing matrix $P = (p_{i,j})_{I\times J}$, which minimizes the total mean queue length of the system in heavy traffic.

$$\min_{P} \quad \tilde{\varphi}(P) = \sum_{j=1}^{J} \frac{\left(\sum_{i=1}^{I} \lambda_i p_{ij}\right)^2 \left(\sum_{i=1}^{I} \lambda_i \vartheta_i p_{ij}\right)}{\sum_{i=1}^{I} p_{ij}} \tag{B.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \rho_i p_{ij} = 1, \quad 1 \le j \le J \tag{B.2}$$

$$\sum_{j=1}^{J} p_{ij} = 1, \quad 1 \le i \le I \tag{B.3}$$

$$0 \le p_{ij} \le 1, \quad 1 \le i \le I, \quad 1 \le j \le J,$$

where the parameters include the mean arrival rate λ_i, the offered load $\rho_i = \lambda_i/\mu_i$ and the second moment of the processing times $\vartheta_i = (\mu_i^{-2} + \beta_i^2)$ for each task type i, $1 \leq i \leq I$.

By analyzing the optimal solution P^*, it is observed that a random routing policy which aims to reduce the total queue length tends to have the following properties.

1. A number of servers are dedicated to processing a task type with high load ($\rho_i > 1$). This is consistent with the intuition that to reduce the mean delay, one means is to reduce the service time variabilities. For task type i, the number of dedicated servers (if any) is equal to $\lfloor \rho_i \rfloor$, so that each server's utilization is one in heavy traffic. There might be more than one group of dedicated servers, each group for a different task type.

2. A number of task types are exclusively processed by one server. On the other hand, the number of task types being processed at each server is small. This also contributes to reducing the service time variabilities.

3. For systems with a large number of servers, there are identical columns in the matrix P^*. Theorem 4.1.2 implies that pooling the corresponding servers can further reduce the total queue length of the system.

B.1 The Procedures

According to the desirable properties described above, we propose in Figure B.1 a heuristic that provides an alternative way to generate a suboptimal routing matrix P^H. It is useful when an NLP solver is not available, or when the time spent in solving NLP (B.1) is very long, or when a large degree of pooling is desirable.

The heuristic starts with a full matrix P^0, with all of its elements being $1/J$. It not only is a natural choice that is a feasible solution to the constraints (B.2) and (B.3), but it also improves the chances of obtaining a load-balanced solution P^H, i.e., for each row i, the values of the nonzero elements $p_{i,j}^H$ are

```
1   function P^H = RoutingMatrix(J, λ, μ, β)
2   {   ρ: array of the offered loads of I task types;
3       ϑ: array of the second moments of the processing times of I task types;
4       P^0: starting point of the routing matrix with size I × J;
5       J^ds: number of the dedicated servers, each processing only one type of tasks;
6       P: the first J^ds columns of the matrix P corresponds to the dedicated servers;
7           If J^ds > 0, φ̃(P) < φ̃(P^0); otherwise, P = P^0;
8       ρ^ds: array of the offered loads sorted in descending order;
9           ρ^ds[i] = 0 implies all of the type i tasks are routed to the dedicated server(s);
10      P^H: the solution returned, with φ̃(P^H) ≤ φ̃(P^ds);
11
12      ρ_ι = λ_ι/μ_ι,    ϑ_ι = (μ_ι^{-2} + β_ι^2),    1 ≤ ι ≤ I;
13      P^0 = (p_{ι,j})_{I×J} with p_{ι,j} = 1/J, for all 1 ≤ i ≤ I, 1 ≤ j ≤ J;
14      [J^ds, ρ^ds, P] = DedicateSrvr(ρ, P^0);
15      ϑ^ds = {ϑ_ι : ϑ_ι ∈ ϑ, ρ^ds[i] ≠ 0};
16      P^H = DedicateType(J^ds, ρ, ϑ^ds, P);
17  }
```

Figure B.1: A heuristic to generate the random routing matrix P^H

close to each other. Since the system in question has identical servers, load balancing is beneficial to reducing the total mean queue length.

Figure B.2 shows the subroutines to generate the dedicated servers. If there exist J^{ds} $(0 < J^{ds} \leq J)$ dedicated servers and the objective function $\tilde{\varphi}(P)$ is decreased, then the resulting matrix P is updated in such a way that the remaining load of each task type is now evenly distributed among the $(J - J^{ds})$ non-dedicated servers. Specifically, we have for all $(J^{ds}+1) \leq j \leq J$,

$$p_{\iota,j} = \begin{cases} 1/(J - J^{ds}), & i \in \{i' : p_{\iota',j'} = 0,\ 1 \leq j' \leq J^{ds}\}, \\ (1 - \sum_{j=1}^{J^{ds}} \rho_{i,j}^{-1})/(J - J^{ds}), & i \in \{i' : p_{i',j'} = \rho_{i',j'}^{-1} > 0,\ 1 \leq j' \leq J^{ds}\}. \end{cases}$$

This keeps the load balanced as much as possible, while maintaining P to be

```
1   function [ J^ds, ρ^ds, P^ds ] = DedicateSrvr(ρ, P)
2   {   col: column index of the routing matrix P with size I × J;
3       r_1, r_2: arrays of row indices of P, with lengths I_1 and I_2 = I − I_1, respectively;
4       k: index of the elements in r_1;
5       update, improve: boolean variables;
6
7       [ ρ^ds, r_1, r_2 ] = Sort1(ρ);
8       P^ds = P;   J^ds = 0;   col = 0;   improve = 1;
9       while (col < J AND improve == 1)
10      {   col = col + 1;   update = 0;   k = 1;
11          while (k ≤ I_1   AND   update == 0)
12          {   P^test = Test1 (col, r_1[k], ρ^ds, P^ds) ;
13              if (φ̃(P^test) < φ̃(P^ds))
14                  P^ds = P^test;   update = 1;
15              else
16                  k = k + 1;
17          }
18          if (update == 1)
19              ρ^ds[ r_1[k] ] = ρ^ds[ r_1[k] ] − 1;   J^ds = J^ds + 1;   [ ρ^ds, r_1, r_2 ] = Sort1(ρ^ds);
20          else   // all the task types with load greater than one are tested.
21              improve = 0;
22      }
23  }
24  // function Sort1(ρ) returns the arrays ρ^ds, r_1 and r_2, where
25          ρ^ds[ r_1[1] ] ≥ ··· ≥ ρ^ds[ r_1[I_1] ] ≥ 1 > ρ^ds[ r_2[1] ] ≥ ··· ≥ ρ^ds[ r_2[I_2] ].
26  // function Test1 (j, i, ρ, P) returns the matrix P^test, which satisfies the constraints
27          (B.2) and (B.3). At the column j, P^test[i, j] > 0 and P^test[i, j] = 0
28          for all i' ≠ i, i.e., the server j is dedicated to processing the task type i.
```

Figure B.2: Subroutines to generate the dedicated servers

```
1   function P^H = DedicateType (J^dt, ρ, ϑ^dt, P)
2   {   col: column index of the routing matrix P with size I × J;
3       t, s: arrays of row indices of P, with the same length K = length(ϑ^dt);
4       k: index from 1 to K; task type s[K] has the smallest processing time variance.
5       update, improve: boolean variables;
6
7       s = Sort2(ϑ^dt, 0);    // ϑ^dt[s[1]] ≥ ··· ≥ ϑ^dt[s[K]].
8       P^H = P;    col = J^dt;    improve = 1;
9       while (col < J AND improve == 1)
10      {       col = col + 1;    update = 0;    k = K;
11              while (k > 0 AND update == 0)
12              {       t = Sort2(ϑ^dt, ϑ^dt[s[k]]);    P^test = Test2(col, s[k], t, ρ, P^H);
13                      if (φ̃(P^test) < φ̃(P^H))
14                          P^H = P^test;    update = 1;
15                      else
16                          k = k − 1;
17              }
18              if (update == 1)
19                  s = s \ {s[k]};    ϑ^dt = ϑ^dt \ {ϑ^dt[s[k]]};    K = K − 1;
20              else
21                  improve = 0;
22      }
23  }
24  // function Sort2(ϑ, ϑ[i]) returns the array r with length K = length(ϑ) such that
25              ‖ ϑ[i] − ϑ[r[1]] ‖ ≥ ··· ≥ ‖ ϑ[i] − ϑ[r[K]] ‖.
26  // function Test2 (j, i, t, ρ, P) returns the matrix P^test. For the dedicated task type i,
27              P^test[i, j] > 0 and P^test[i, j'] = 0 for j < j' ≤ J. Moreover, I' task types
28              (indexed by t[1]...t[I']) are routed to the servers j' instead of the server j,
29              while the number I' is subject to the constraints (B.2) and (B.3).
```

Figure B.3: Subroutines to generate the dedicated task types

a feasible solution.

Figure B.3 gives the routines to generate the dedicated task types. Suppose there are I_j^{dt} types of tasks being processed exclusively at non-dedicated server j. The variabilities of their processing times (indicated by ϑ_i, $1 \leq i \leq I_j^{dt}$) are close to each other. The load of the remaining $(I - I_j^{dt})$ task types is then evenly distributed among the non-dedicated servers which are indexed by $j' \in \{(j+1), \ldots, J\}$. (In Figure B.3, $(I - I_j^{dt})$ is denoted by I' in **function** `Test2`.)

B.2 Numerical Examples

Here we give an example of computing the routing matrix P^H using the heuristic. For comparison purposes, this example has the same parameters as in Example 4.1.1.

Example B.2.1. *Let $I = 5$ and $J = 6$. The first-order primitives are*

$$\lambda = \begin{bmatrix} 39.84 & 13.47 & 15.15 & 0.94 & 2.06 \end{bmatrix}$$

and

$$\mu = \begin{bmatrix} 16.7 & 30.4 & 18.9 & 3.0 & 1.0 \end{bmatrix}.$$

So the offered loads are given by the vector

$$\rho = \begin{bmatrix} 2.39 & 0.44 & 0.80 & 0.31 & 2.06 \end{bmatrix}.$$

Since the processing times are assumed exponentially distributed, the second moments are given by the vector

$$\vartheta = 2\mu^{-2} = \begin{bmatrix} 7.2 & 2.2 & 2.6 & 222.2 & 2000.0 \end{bmatrix} \times 10^{-3}.$$

Using the heuristic, the suboptimal solution to NLP (B.1) *is obtained to be*

$$P^H = \begin{bmatrix} 0.42 & 0.42 & 0.06 & 0.10 \\ & & 1.00 & \\ & & 0.50 & 0.50 \\ & & & 1.00 \\ 0.49 & 0.49 & & 0.02 \end{bmatrix}, \tag{B.4}$$

which yields the objective function value

$$\tilde{\varphi}_n^H = 234.8. \tag{B.5}$$

By comparing (B.5) with $\tilde{\varphi}_{exp}^* = 178.6$ in (4.33), the relative error of the heuristic is

$$\varepsilon = \frac{\tilde{\varphi}_n^H - \tilde{\varphi}_{exp}^*}{\tilde{\varphi}_{exp}^*} \times 100\% = 31\%.$$

On the other hand, by comparing (B.4) with (4.32), it is found that P^H has more identical columns than P^*. By (partial) pooling the servers 1 and 3, 2 and 4, respectively, we have the objective function value

$$\tilde{\varphi}_p^H = 150.6,$$

which is 16 percent less than the minimum value $\tilde{\varphi}_{exp}^*$ for no pooling.