Queueing-Theoretic Approaches for Dynamic Scheduling: A Survey

Daria Terekhov[a], Douglas G. Down[b] and J. Christopher Beck[a]
[a]Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, Canada
{dterekho,jcb}@mie.utoronto.ca,
[b]Department of Computing and Software, McMaster University, Canada
downd@mcmaster.ca

Within the combinatorial scheduling community, there has been an increasing interest in modelling and solving scheduling problems in dynamic environments. Such problems have also been considered in the field of queueing theory, but very few papers take advantage of developments in both areas, and literature surveys on dynamic scheduling usually make no mention of queueing approaches. In this paper, we provide an overview of queueing-theoretic models and methods that are relevant to scheduling in dynamic settings. This paper provides a context for investigating the integration of queueing theory and scheduling approaches with the goal of more effectively solving scheduling problems arising in dynamic environments.

Key words: Dynamic Scheduling, Queueing Theory

---

1 Introduction

Real-world scheduling problems are combinatorial, dynamic and stochastic. The goal in such problems is to determine an approach that dictates, at every decision epoch, how the available resources should be allocated among competing job requests in order to optimize the performance of the system. The combinatorial scheduling literature (Graham et al., 1979; Pinedo, 2003; Leung, 2004; Pinedo, 2009; Baker and Trietsch, 2009) has mostly focused on solving static and deterministic versions of these problems, and views a dynamic scheduling problem as a collection of linked static problems (Bidot et al., 2009; Ouelhadj and Petrovic, 2009; Aytug et al., 2005; Davenport and Beck, 2000; Suresh and Chaudhuri, 1993). Thus, methods developed for static scheduling problems become directly applicable to dynamic ones. Such methods can effectively deal with complex combinatorics and can optimize the quality of static sub-problem schedules, but tend to overlook the long-run performance and the stochastic properties of the system, with notable exceptions being the work on anticipatory scheduling (Branke and Mattfeld, 2002, 2005) and online stochastic combinatorial optimization (Van Hentenryck and Bent, 2006).

Scheduling	Queueing
A survey of dynamic scheduling in manufacturing systems (Ouelhadj and Petrovic, 2009)	*Queueing theory in manufacturing: A survey* (Govil and Fu, 1999)
Scheduling and control of flexible manufacturing systems: a critical review (Basnet and Mize, 1994)	*Flexible manufacturing systems: a review of analytical models* (Buzacott and Yao, 1986)
Job shop scheduling techniques in semiconductor manufacturing (Gupta and Sivakumar, 2006)	*Queueing theory for semiconductor manufacturing systems: A survey and open problems* (Shanthikumar et al., 2007)

Table 1: Pairs of complementary scheduling and queueing papers that focus on manufacturing.

It has long been recognized that queueing theory can also provide a basis for examination of scheduling problems (Conway et al., 1967). Moreover, the queueing theory literature frequently considers the same application areas as scheduling. For example, both have extensively studied manufacturing environments, as is demonstrated by the complementary papers listed in Table 1.

To our knowledge, only a few papers (e.g., those by Nazarathy and Weiss (2010), Bertsimas and Sethuraman (2002) and Bertsimas et al. (2003)) combine queueing and scheduling ideas to address static scheduling problems, and only two recent dissertations (Tran, 2011; Terekhov, 2013), and a subsequent paper (Terekhov et al., 2014), propose methods for dynamic scheduling that take advantage of developments in both of these areas. It is true that scheduling books such as those by Leung (2004) and Chrétienne et al. (1995) have chapters on both deterministic scheduling and queueing approaches, but they usually make no link between queueing theory and the dynamic scheduling frameworks developed by the combinatorial scheduling community.[1] Additionally, except for the work of Suresh and Chaudhuri (1993), literature surveys on scheduling in dynamic environments make no mention of queueing approaches. This characteristic of the scheduling literature may be partially due to the fact that a significant proportion of queueing research has dealt with the development of descriptive models for evaluation of the long-run expected behaviour of a system, while scheduling is prescriptive in nature and frequently considers only short-run performance measures. Nevertheless, there has also been a substantial amount of work on prescriptive queueing models that aims to provide a policy for stating which job or job class should be processed next.

The objective of this paper is to provide a review of the work done by researchers in queueing theory that is relevant to scheduling, and to provide a foundation for the investigation of the integration of queueing theory

[1] It is important to note that in early research on resource allocation and sequencing, queueing theory and scheduling were more unified. For example, the book *Theory of Scheduling* by Conway et al. (1967) contains many fundamental results for both deterministic scheduling problems and queueing problems.

and scheduling methods for solving dynamic scheduling problems. The literature on scheduling in the context of queueing systems is extensive, and so our review is by no means exhaustive: our goal is to provide the reader with a high-level view of the relevant material. We present the basic mathematical representation for each of the methodologies covered in our review in order to provide the reader with a better understanding of each approach and of the differences among the approaches. Some methodologies, such as those in Section 4.3, are more complex than others, and require substantial mathematical notation and/or statement of assumptions. For a more mathematically rigorous coverage of scheduling models developed within queueing theory we refer the reader to the comprehensive book by Meyn (2008).

Throughout the review, we make connections between the queueing theory and the scheduling literatures. We use the term *scheduling* to mean the process of making decisions which specify the order in which the jobs should be processed on the given machines (i.e., job sequencing or operational-level scheduling). We therefore do not review the related issues of queueing control and design (Tadj and Choudhury, 2005; Crabill et al., 1977; Govil and Fu, 1999) such as admission control (control of arrival rates, decision to accept/reject an arriving job, appointment scheduling) (Stidham, 1985; Fan-Orzechowski and Feinberg, 2007; Hajek, 1984; Pegden and Rosenshine, 1990), routing control (Ephremides et al., 1980; Veatch and Wein, 1992; Gurvich and Whitt, 2009), server assignment (Ahn et al., 2002; Andradóttir et al., 2003) and control of service rates (Weber and Stidham, 1987; Grassmann et al., 2001). Routing control and server assignment problems discussed in the queueing literature are resource allocation questions that have a substantial influence on the construction of schedules in dynamic environments with multiple machines. In server assignment models, a rule for assigning machines to job types needs to be determined and decisions are typically made when a machine becomes free; in routing models, on the contrary, machines have their own queues and the assignment of a job to a particular machine needs to be made at the arrival time of the job (Righter, 1994). Investigating the relationship between queueing design and control models, and scheduling is worthwhile and interesting, but a single paper is not sufficient to cover all of the relevant connections and thus we leave such an investigation for future work.

Since our focus is on queueing-theoretic approaches for scheduling, we do not discuss work on the use of other techniques to solve problems that can be described in a queueing setting, such as sample path analysis (El-Taha and Stidham, 1999; Robinson, 1996; Plambeck et al., 1996), simulation and mixed-integer programming (Chan and Schruben, 2008), and approaches from the electrical engineering and control literature (e.g., model predictive control (Garcia et al., 1989; van Leeuwaarden et al., 2010), linear and affine controllers (Boyd and Barratt,

1991; Skaf and Boyd, 2010)). Future work should investigate the connections between the use of these methods in queueing and scheduling settings and, in particular, compare sample path and model-predictive methods with predictive-reactive (Bidot et al., 2009) and online stochastic combinatorial optimization approaches (Van Hentenryck and Bent, 2006) from the scheduling literature. The work on robust queueing (Bertsimas et al., 2011; Bandi and Bertsimas, 2012; Bandi et al., 2012) falls outside of the scope of our paper as well, since it replaces stochastic process primitives with uncertainty sets and thus is not a traditional queueing-theoretic approach. To the best of our knowledge, robust queueing has, to date, focused on descriptive (performance analysis) problems, whereas our main interest is in prescriptive models dealing specifically with scheduling (see Section 3.3 for a discussion of descriptive and prescriptive queueing models). We also do not review methods from the following areas: stochastic scheduling, which focuses on problems with a fixed set of jobs that have stochastic characteristics (Pinedo, 2003; Baker and Trietsch, 2009); online scheduling (Pruhs, 2007), which studies competitive analysis and worst-case performance in dynamic scheduling problems without distributional knowledge; and real-time scheduling, which addresses problems that have explicit timing (i.e., deadline) requirements, which may be deterministic or probabilistic (Sha et al., 2004; Lehoczky, 1996).

We restrict our attention to unary-capacity single and multiple machine scheduling problems with jobs that arrive dynamically over time. We assume that all of the parameters necessary for defining a queueing system (see Section 3) that cannot be modified by the decision-maker are known with certainty, although not all of the scheduling approaches surveyed use all of the available parametric information to make a decision. The reader who is interested in scheduling of queueing systems with arrival and processing rates being modelled by uncertainty sets is referred to the dissertation by Su (2006). Su (2006) applies the robust optimization framework of Bertsimas et al. (2004) and Bertsimas and Sim (2004) to the fluid model representation of a queueing system, which is discussed in Section 4.3.1.1. For a game perspective, where the uncertain parameters are assumed to be chosen by an adversary, we refer to the work of Day (2006) and Dupuis (2003). For an overview of queueing models for systems with interruptions, we refer to the paper by Wu (2014).

The current paper is organized as follows. In Section 2, we describe our classification of scheduling problems. Section 3 provides a brief introduction to queueing theory and general queueing models, making the distinction between single-station and multi-station models, between single-class and multi-class systems, and between descriptive and prescriptive models. In Section 4, a variety of queueing-theoretic approaches that may be helpful for making operational-level scheduling decisions are surveyed. The section is divided into three parts: Markov

Decision Processes (MDPs), which allow for investigation of optimal scheduling policies without assuming a particular policy structure; queueing models that aim to optimize decision-making when a particular policy type is assumed; and models based on alternative representations (e.g., approximations). Such an organization is aimed at providing an understanding of the main methodological streams in queueing theory that address scheduling problems. Within each of the three parts, we discuss both single-machine and multiple-machine scheduling problems. Section 5 summarizes future work ideas for integrating queueing theory and scheduling that are mentioned throughout the review. We conclude in Section 6.

2 Classification of Scheduling Problems

In a static and deterministic environment, scheduling corresponds to determining the start times of a given set of jobs that has to be processed on one or several machines. This problem is combinatorial in nature, and involves a finite set of jobs with known characteristics and a finite time period whose length is equivalent to the schedule's makespan. In reality, such a problem corresponds to only one possible scenario that the scheduler may be faced with at a particular point in time, and the schedule created based on this scenario is likely to be valid for only a short horizon. In fact, the scenario represents a realization of many stochastic processes that govern the evolution of the system in question (e.g., inter-arrival times, processing times, etc.). From the perspective of real scheduling problems, static and deterministic scheduling deals with short-run decisions based on a local, myopic view of the environment, its combinatorial structure and realized uncertainty. The majority of the classical scheduling literature deals exactly with such problems.

A static and stochastic problem aims to determine the order in which a fixed set of jobs with stochastic characteristics should be processed. Such a problem deals with a short time horizon and adopts a myopic view of the system, corresponding to a particular realization of the job arrival process at a specific point in time.[2] Thus, the problem models a variety of scenarios for a particular set of jobs, and the schedule has to be constructed in a way that deals not only with the combinatorics of the problem, but also with its stochastic nature: since it is impossible to create a schedule that would be of high quality in all scenarios, the goal in such problems is usually to achieve good (or optimal) performance in a probabilistic sense (e.g., in expectation). Work on these problems is known as stochastic scheduling (e.g., see the books by Pinedo (2003) and Baker and Trietsch (2009)), and methods for stochastic scheduling are based on approaches for deterministic problems as well as some probabilistic reasoning

[2]The realization may be partial in the sense that, for the given set of jobs, the actual arrival times (release dates) of jobs might not be known with certainty.

(Righter, 1994; Beck and Wilson, 2007; Daniels and Carrillo, 1997; Wu et al., 2009; Leon et al., 1994; Davenport et al., 2001; Herroelen and Leus, 2005).

The problem of scheduling in a dynamic environment involves a long time horizon and has to deal with all the possible realizations of the job arrival process and of the job characteristics. The ultimate goal in solving this problem is to construct a schedule that is optimal for the specific realization that actually occurs. The quality should be close to that of the schedule that could have been constructed if all of the uncertainty had been revealed *a priori*. Clearly, this is a difficult task, because to make a decision, one can use only the information that is known with certainty at that particular decision point and the stochastic properties of scenarios that can occur in the future. In addition, the effect of the decision on both short-run and long-run performance has to be considered. Interestingly, "the problem of planning or scheduling dynamically over time, particularly planning dynamically under uncertainty" was the initial motivation for George Dantzig's research on linear and mathematical programming; writing in 1991, Dantzig stated that "If such a problem could be successfully solved it could eventually through better planning contribute to the well-being and stability of the world" (quoted in Ben-Tal et al. (2009)).

To deal with dynamic scheduling problems, queueing theory and scheduling have adopted different approaches. Queueing theory has taken the viewpoint that, since it is impossible to create an optimal schedule for every single sample path in the evolution of the system, one should aim to achieve optimal performance in some *probabilistic* sense (e.g., in expectation) over a long time horizon. This goal could be attained by construction of a policy based on the global stochastic properties of the system. For example, a policy could specify how start time decisions should be made each time a new job arrives or a job is completed. The schedule resulting from such a policy, while being of good quality in expectation, may be far from optimal for the particular realization of stochastic processes that occurs. Moreover, queueing theory generally studies systems with simple combinatorics, as such systems are more amenable to rigorous analysis of their stochastic properties, and usually assumes distributional, rather than exact, knowledge about the characteristics of jobs or job types.

In the scheduling community, a dynamic scheduling problem is generally viewed as a collection of linked static problems. This viewpoint implies that methods developed for static scheduling problems become directly applicable to dynamic ones. Such methods can effectively deal with complex combinatorics and can optimize the quality of the schedules for each static sub-problem. Except for the work on anticipatory scheduling (Branke and Mattfeld, 2002, 2005) and online stochastic combinatorial optimization (Van Hentenryck and Bent, 2006), scheduling methods tend to overlook the long-run performance and the stochastic properties of the system.

Thus, the perspectives taken by queueing theory and scheduling with respect to dynamic scheduling problems are complementary: the former can effectively deal with the dynamism and stochasticity in the system, while the latter is able to reason about the combinatorics.

3 Queueing Theory Fundamentals

Queueing theory can be defined as the mathematical study of waiting lines (queues) (Gross et al., 2008). It therefore models systems in which one or more servers (machines) at one or more service stations serve (process) arriving customer requests (jobs).[3] As stated by Gross et al. (2008), a mathematical model of a queue is composed of six main features.[4]

Arrival Pattern The main characteristic of the arrival pattern is the probability distribution of times between successive customer arrivals. This distribution may or may not be stationary (independent of time) and may or may not depend on the number of customers in the queue. Customers may arrive one-by-one or in a group (batch) whose size may be determined by a probability distribution (which could be deterministic). A customer may decide to join the queue and stay there until receiving service, join the queue but leave without service (renege) due to a long wait, or not join the queue at all (balk). If the system consists of more than one queue, a customer may decide which queue to join and/or to switch (jockey) from one queue to another.

Service Pattern Similarly, the service pattern is characterized by a service time[5] distribution, which may be stationary or non-stationary and which may or may not depend on the number of customers waiting for service or their waiting time. Customers may be served singly or in groups whose size is determined by a probability distribution (which could be deterministic). The service time of a customer may become known with certainty upon arrival or at the end of service.

Queue Discipline The queue discipline is a rule that determines the order in which customers receive service. The simplest and most common queue discipline is first-come, first-served (*FCFS*). Other examples of queue dis-

[3] The terms *server* and *customer* are more common in the queueing literature, while the terms *machine* and *job* are more common in scheduling. In this section on queueing theory fundamentals, we use the queueing terminology. For the rest of the review, we employ scheduling terms, except when the description clearly demands otherwise (e.g., in Section 4.2.2, which talks about polling models).

[4] Throughout the paper, we take many of the fundamental queueing theory concepts and results from the book by Gross et al. (2008), an excellent introduction to queueing theory. The reader is also referred to the textbook by Kleinrock (1976).

[5] The scheduling literature employs the term *processing time* instead.

ciplines include last-come, first-served (*LCFS*), random selection for service (*RSS*) or shortest processing time (*SPT*) first. The queue discipline is, essentially, a scheduling policy, and the reader may note the strong similarity between these examples of policies and dispatching rules in the scheduling literature (Pinedo, 2009).

System Capacity A queueing system may have finite or infinite capacity. In a finite capacity queue, there is a limit on the number of customers that can be present in the system at any point in time, and customers who arrive at times when this limit has been reached are not allowed to enter the system.

Number of Service Stations and Servers A queueing system may be composed of one or more stations (stages), each of which has one or more servers (channels). A multi-station system may be thought of as a network of interconnected nodes, with each node consisting of one or more queues with one or more servers. When there are several servers at a station, there may be a queue for each server, as in a supermarket, or one queue for all servers, as in the case of a bank. In a multi-station system, the order in which customers visit the stations (the routing) may be deterministic or stochastic. The distinction between single-station and multi-station models is discussed in more detail below.

Additional characteristics of queueing models include the server types (see, for example, the work on flexible servers by Ahn et al. (2002), Andradóttir et al. (2003) and Gurvich and Whitt (2009)) and the customer types served by a system, as discussed in Section 3.2. Such characteristics affect the kinds of scheduling problems that need to be solved.

3.1 Single-station vs. Multi-station Models

In this section, we give some additional characteristics of single-station and multi-station queueing models and their relation to single-machine and multi-machine scheduling problems.

Single-station Queueing Models In the scheduling literature, the notation used for describing a scheduling problem is $\alpha|\beta|\gamma$, where α represents the machine environment (e.g., the number of machines and the relations among them), β represents job processing characteristics and constraints, and γ is the objective function (Graham et al., 1979; Pinedo, 2003). Similarly, in queueing theory a single-station queueing model is usually specified by a combination of symbols $A/B/X/Y/Z$, where A and B specify the inter-arrival and service time distributions,

respectively, X is the number of parallel servers, Y indicates the capacity of the system and Z describes the queue discipline. Typical entries for A and B are: D, which stands for deterministic[6]; M, which stands for an exponential distribution (Markovian); GI, which stands for a general independent distribution; or G, which corresponds to a general[7] distribution. X and Y are represented by positive integers or ∞. Examples for Z include *FCFS*, *LCFS*, *RSS*, priority (*PR*) or general discipline (*GD*). When the Y or Z fields are empty, infinite capacity and *FCFS* discipline are assumed, respectively. For example, an $M/G/3/5/LCFS$ queue is a queue with Markovian arrivals (M), a general service time distribution (G), three servers, a system capacity of five jobs, and last-come, first-served order of service.

Assuming equivalence between a server in queueing and a machine in scheduling, a single-station model corresponds either to a single-machine or a parallel-machine scheduling environment, depending on the number of servers. If there are multiple machines, having one queue in the system implies that any job can be processed on any machine, while having a distinct queue attached to each server models the situation when each job has to be processed on one specific machine.

Multi-station Queueing Models A queueing system with more than one service stage is typically referred to as a queueing network or a network of queues. It consists of a set of nodes, each of which is a (single-stage) queue. Jobs typically require service at several nodes (stations) of the network in some order (routing).

Queueing networks in which jobs enter the system "from the outside" and leave the system upon receiving service are referred to as open queueing networks. Models in which a finite population of jobs circulates within the system, and there are no arrivals from or departures to the outside, are called closed queueing networks. Semi-open networks include characteristics of both open and closed networks (Chen and Yao, 2001), i.e., there are arrivals from the outside but the system has fixed capacity. Definitions of such networks in the literature differ mainly with respect to what happens to arriving customers when the system's capacity has been reached. As noted by Massey and Srinivasan (1991), some authors (Dubois, 1983; Chen and Yao, 2001) assume that arriving customers are blocked and lost when the system is full, while others (Buzacott and Shanthikumar, 1985; Jia and Heragu, 2009) assume that arriving customers wait outside and enter the system when space becomes available. Massey and Srinivasan (1991) assume that a semi-open system is divided into parts, such that the numbers of jobs in some regions of the network are controlled, while in others are unconstrained. Open networks in which jobs can

[6]Deterministic refers to all inter-arrival or service times having the same value.

[7]For a general distribution, there are no assumptions regarding its precise form. Thus, results that apply to the general distribution apply to any specific distribution.

arrive at a node only from the outside or from an upstream node are referred to as acyclic or feed-forward networks. This kind of network can be analyzed in a recursive manner, starting from the upstream stations. In networks with feedback, on the contrary, jobs may visit the same station more than once (Chen and Yao, 2001). More complex networks may involve fork-join queues, in which an arriving job is split into sub-jobs that have to be processed in parallel and then reassembled (Bose, 2002; Baccelli et al., 1989). The work of Baccelli and Makowski (1989) and Baccelli and Liu (1990) considers synchronized queueing networks, which can model multiprocessor systems that process programs consisting of multiple tasks related by precedence constraints.

It has long been recognized that queueing networks are good models for dynamic job shop environments (Jackson, 1963; Wein and Chevalier, 1992; Buzacott and Shanthikumar, 1985, 1993). Semi-open and open network models provide the most intuitive representations of dynamic job shops since they allow one to explicitly model the process of job arrivals. A semi-open network can be used to represent a job shop with a strict limit on the number of jobs in the shop, while an open network can model a system with no such limit. Closed queueing networks can be used to model job shops in which the total number of jobs remains constant or, in other words, for which it is valid to assume that a new job arrives at the same instant as another job is completed and leaves. Examples of environments that can be modelled as closed queueing networks include production systems that have a constant work-in-process inventory or follow a one-for-one replenishment policy (base-stock control rule) (Chen and Yao, 2001). Alternatively, one can imagine that there is a fixed number of palettes circulating through the service stations of the system, that raw materials are placed on one such palette and that these raw materials are gradually transformed into finished products as they receive processing (Williams, 1996). Acyclic networks can represent flow shops while networks with fork-join queues can be used to model shops with assembly operations. Synchronized queueing networks can be useful for machine scheduling problems with complex precedences among the activities, such as the task management problem (Myers et al., 2007), and for project scheduling.

A generalization of a queueing network is referred to as a stochastic network (Kelly et al., 1996) or a stochastic processing network (Harrison, 1996). A stochastic processing network is defined by a set of buffers, a set of activities and a set of processors. Buffers hold jobs that have arrived to the system and are awaiting processing. Each activity uses one or several processors in order to process one or several jobs (which may belong to different buffers).[8] Scheduling in stochastic processing networks corresponds to the determination of the order of executing

[8]This definition of activity can be viewed as a generalization of the notion of activity in scheduling: it represents a particular processing task. Classical scheduling typically assumes that the task is performed on one job by one machine, whereas stochastic processing networks allow modelling of the case when one task corresponds to multiple job components being processed by multiple machines simultaneously (e.g., assembled by a team of workers).

the different activities on the various available processors. Stochastic processing networks can be used to model more complex scheduling environments than those that can be modelled by multi-class queueing networks. For example, they can allow one to represent material handling and machine-operator interaction (Dai and Lin, 2005) or input-queued switches (Dai and Prabhakar, 2000). In the remainder of this literature review, we focus on queueing networks rather than stochastic processing networks.

3.2 Single-class vs. Multi-class Systems

In the literature, a distinction is usually made between single-class and multi-class queueing networks. In a single-class network, jobs being processed or waiting for processing at any given station are assumed to be indistinguishable. A multi-class system is one in which several classes of jobs are served at each station (Posner and Bernholtz, 1968; Harrison and Nguyen, 1993). A class is usually defined as a combination of job type and processing stage. An instance of a job belonging to a particular class is equivalent to an activity or operation in scheduling terminology. The reader should note a slight subtlety arising from these definitions. Specifically, in a single-class network, according to the definition of a class, there are, in total, as many classes as there are stations (Harrison and Nguyen, 1993). The term *single*-class refers to the fact that *a single class is served at each station*. A multi-class network is called mixed when it is open for some classes of jobs and closed for others (Bose, 2002).

It is important to note that the individual jobs belonging to the same class are different: they have unique arrival times, unique processing times and, in the case of multiple machines, may have a unique route. However, these differences between the jobs are stochastic variations – all jobs in a class are governed by the same stochastic processes and are, except for these stochastic variations, indistinguishable.

In a single-class system, scheduling corresponds to determining the order in which the jobs should be processed at each node of the network. Since the jobs are stochastically indistinguishable, scheduling decisions have to be based on realizations of job characteristics such as their processing times. In the context of single-class systems, the queueing literature mostly focuses on the performance evaluation of scheduling policies (e.g., *FCFS*, *LCFS*, shortest-remaining-processing-time-first, processor-sharing) (Shanthikumar, 1982; Wolff, 1989; Harchol-Balter, 2011). For instance, one very strong result is the optimality of the shortest-remaining-processing-time policy in an $M/G/1$ queue (Schrage, 1968; Smith, 1978).[9]

[9]Not surprisingly, for the single-machine case the scheduling literature has analogous results: for a static deterministic problem with n jobs, the shortest-processing-time-first rule minimizes the total flow time; the weighted-shortest-processing-time rule minimizes the total weighted flow time (equivalently, the sum of weighted completion times) (Pinedo, 2003; Baker and Trietsch, 2009). Similarly, Baker and Trietsch (2009) and Pinedo (2003) show that, for a problem with n jobs and stochastic processing times (i.e., the realization of the processing time for a given operation is not known until it is finished), the shortest (weighted) expected-processing-time rule is optimal for the (weighted) expected flow

Scheduling in multi-class queueing networks involves the determination of a policy that specifies which class of jobs should be processed next on each machine. This problem falls under the category of models for control of multi-class queueing networks, which are known to be mathematically challenging (Bertsimas et al., 1994). In particular, the choice of job to be processed next at every station of the network at every decision time point may depend not only on the number and characteristics of jobs present in the station's queue, but also on the state of the other nodes in the network.

3.3 Descriptive vs. Prescriptive Models

Queueing theory is composed of work on descriptive and prescriptive (control) models (Gross et al., 2008; Stidham, 2002). The goal of descriptive queueing theory is to evaluate the performance of a queueing system based on some assumptions about its characteristics. Typical performance measures (Adan and Resing, 2002) include

- the distributions of the amount of time spent by jobs in the system (sojourn time) and of the jobs' waiting time in the queue prior to receiving service (queueing time),
- the distributions of the number of jobs in the system and in the queue,
- the distribution of the amount of work in the queue, with work being defined as the sum of the service times of the jobs waiting in the queue and the residual service time of the job(s) currently receiving service,
- the distribution of the length of the busy period of a server, which is the time period during which the server is continuously busy,
- cost-based measures such as expected holding costs (Reiman and Wein, 1998) or net present value of the difference between rewards and costs over an infinite time horizon (Harrison, 1975).

Most of descriptive queueing theory focuses on steady-state analysis of the system, that is, its performance over a long period of time during which the system behaviour should stabilize. Specifically, steady-state expected values of the above-mentioned distributions are essential for understanding the performance of the system. Transient analysis (Grassmann, 1977; Kaczynski et al., 2011) and evaluation of time-dependent probability distributions (Massey and Whitt, 1998) provide additional insights into system performance but are, in general, more difficult to derive and hence rarer.

time objective.

The area of queueing theory that deals with prescriptive models is frequently called the *design and control of queueing systems* (Tadj and Choudhury, 2005; Gross et al., 2008). In both queueing design and control problems, the goal is to find optimal values for the *controllable* parameters of the queue. These parameters include the number of machines (channels) available for processing arriving jobs, the limit on the length of the queue(s) (system capacity), the arrival rate of jobs to the queue(s), the service rates of the machine(s), as well as any combination of these. Queueing design problems are static – once the optimal value of a controllable parameter is determined, it becomes a fixed characteristic of the queue. Queueing control problems, on the contrary, are dynamic – the goal in such problems is usually to determine an optimal action to take when the system is in a particular state. For example, consider a retail facility with workers who have to serve stochastically arriving customers and also perform back room tasks which are independent of the customer arrival process (Terekhov et al., 2009). In order to optimize the performance of such a facility, one has to solve the queueing design problem of finding the optimal number of cross-trained servers to employ as well as the related queueing control problem of determining when to dynamically switch these workers between the two task types. Overviews of design and control problems involving queues can be found in the papers of Tadj and Choudhury (2005) and Crabill et al. (1977), and the books by Kitaev and Rykov (1995), Stidham (2009) and Meyn (2008); the reader is also referred to the queueing design and control papers cited in the introduction.

The queueing discipline of each buffer determines the order in which arriving jobs are processed. The queueing discipline can, in fact, be seen as a scheduling policy. Consequently, both descriptive and prescriptive queueing models can be of use in scheduling. Descriptive models can be helpful for analyzing the performance and deriving theoretical properties of particular scheduling policies. Prescriptive models, on the contrary, allow one to determine good or optimal scheduling rules. Some authors (Crabill et al., 1977) classify such models as part of the queueing control literature.

4 Methodologies for Scheduling

This section is aimed at providing the reader with an understanding of the main methodological streams in queueing theory that address scheduling problems. To achieve this goal, we classify queueing methodologies related to scheduling into three categories. Firstly, we discuss Markov Decision Processes (MDPs), which can provide the basis for proving theoretical properties of policies as well as for computation of policy parameters. Theoretically, the MDP approach does not need to place a-priori restrictions on the space of scheduling policies that is consid-

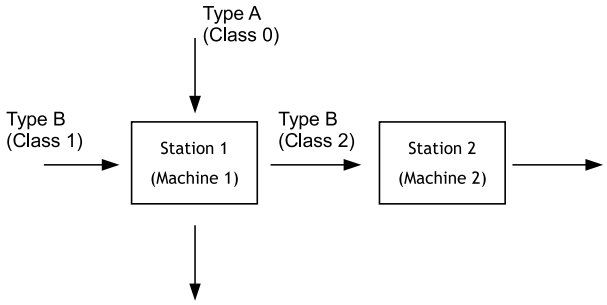

Figure 1: The criss-cross network.

ered. However, when the complexity of the problem grows, MDP approaches fail due to the size of this space. In general, there are two ways to deal with such complexity: optimize within a restricted policy space or solve alternative representations of the problem (such as approximations) to obtain guidance for scheduling decisions in the original problem. These are the remaining two categories we discuss. Within each of the three parts of this section, we make connections with single-machine and multiple-machine scheduling problems from the classical scheduling literature, and provide ideas for future work on the integration of queueing and scheduling.

Of main interest to us are models in queueing theory that can help with sequencing decisions. For illustration purposes, we use a system that is frequently called the *criss-cross* network (Martins et al., 1996), which processes two types of jobs as shown in Figure 1. Jobs of type A require processing at station 1 while jobs of type B need to be processed at station 1 and then at station 2. Each station consists of exactly one machine. There are, therefore, three classes: class 0 corresponds to type A jobs, class 1 to type B jobs at machine 1, and class 2 to type B jobs at machine 2. We assume that class 0 jobs arrive to the system with rate λ_0 and a general inter-arrival distribution; class 1 inter-arrival times also follow a general inter-arrival distribution, with rate λ_1. The arrival rate to class 2 depends on the arrival and processing rates of class 0 and 1, and the scheduling policy employed at machine 1. Processing times for class k are generally-distributed with rate μ_k. This problem setting has been extensively studied in the queueing literature as it captures the key difficulties arising in dynamic scheduling and control (Chen et al., 1994). The reader is encouraged to think about the scheduling and resource allocation questions that may arise in this environment, and about how to address them using a predictive-reactive method or other dynamic scheduling methods. In our illustrative example, we focus on the problem of sequencing type A and B jobs at machine 1 in order to minimize the discounted total cost over an infinite time horizon, assuming that per time unit

holding cost of c_k is incurred for jobs in class k.

Throughout the review, we use consistent notation to describe the different approaches from the literature, rather than using the notation of the respective papers. We use bold-face upper-case letters to denote matrices and bold-face lower-case letters to denote vectors. Comparison of two vectors is always assumed to be component-wise, e.g., $\mathbf{x} \geq \mathbf{y}$ denotes $x_k \geq y_k \ \forall k$. In a few instances, when there is clearly no ambiguity, we re-use notation. We denote the set of classes in a multi-class network by $\mathcal{K}$ and the set of stations[10] by $\mathcal{M}$, with $\sigma(k)$ representing the machine that is required to process class k. $\mathcal{K}_m$ is the set of classes processed by machine m, and $\mathbf{C}$ is the constituency matrix, with each entry (m, k) equal to 1 if $k \in \mathcal{K}_m$ and 0 otherwise. We number classes starting from 0 and machines starting from 1. The arrival and processing rates for class k are denoted λ_k and μ_k, respectively; the corresponding vectors are $\boldsymbol{\lambda} = (\lambda_0, \ldots, \lambda_{|\mathcal{K}|-1})^T$ and $\boldsymbol{\mu} = (\mu_0, \ldots, \mu_{|\mathcal{K}|-1})^T$, where ^{T} denotes transpose. The queue length of class k at time t is denoted by $q_k(t)$, the queue length process for class k by $Q_k = \{q_k(t), t \geq 0\}$, and $\mathbf{q}(t) = (q_0(t), q_1(t), ..., q_{|\mathcal{K}|-1}(t))$. The control policy is typically represented using the letters u and U, although the exact interpretation varies from one methodology to another.

4.1 Markov Decision Processes

An MDP model consists of states, actions, decision points, costs, transition probability distributions and an optimization criterion (Sennott, 1999). In the scheduling literature, it is typical to assume that jobs belonging to the same class are distinguished from each other by their weights, processing times and due dates; schedules are constructed based on these individual characteristics. The MDP approach is general enough to represent scheduling problems under such assumptions, with actions corresponding to the choice of the specific job to process next, or the choice to remain idle. However, the state representation in the corresponding MDP model would need to include all of the known job characteristics and to keep track of the processing sequence, leading to an enormous state space that would make the problem intractable to solve.

The queueing representation of scheduling problems is much more amenable to analysis via MDP methods, since it assumes that jobs belonging to a particular class are stochastically indistinguishable and scheduling decisions amount to choosing the class of jobs to be processed next, rather than the individual job. If the queue discipline is *FCFS*, a state can be defined as the number of jobs of each class present in the system; decision points can be job completion epochs or time points when there is a job arrival; an action may be the choice to process a

[10]Unless otherwise specified, we assume each station has exactly one unary-capacity machine and thus we sometimes use the terms of *station* and *machine* interchangeably.

particular job class on a given machine or for this machine to remain idle (Harrison, 1975). The goal is to find a policy that specifies the action to be taken in every state of the system so that an objective is optimized.

The policies may or may not depend on the history (past states) of the system, or on the actual time point when a decision is made. Derman (1970) classifies MDP policies in a hierarchical fashion. The most general is the set $\mathcal{C}$ of all possible policies, i.e., policies which may be dependent on the complete history of the system. An important subset is $\mathcal{C}_M$, which consists of all memoryless, or Markovian, policies. In such policies, it is assumed that the probability of taking action a is a function of only the current state and the current time. $\mathcal{C}_S$ is a subset of $\mathcal{C}_M$ which consists of time-invariant policies. Under these policies, the probabilities of taking a particular action are dependent only on the system state. A special subset of $\mathcal{C}_S$, $\mathcal{C}_D$, consists of all deterministic policies. A deterministic policy is one in which the probability of taking a particular action a in a state i is either 0 or 1.

We now give a formal definition of a general MDP with a countable state space $\mathbb{X}$ and action space $\mathcal{A}$ based on the paper by Chen and Meyn (1999). For each state $x \in \mathbb{X}$, there is a non-empty subset $\mathcal{A}(x) \subseteq \mathcal{A}$, which consists of actions that are admissible when x is the state at time t, denoted $x = \Phi(t)$. Transitions in the state process Φ occur according to conditional probability distributions $\{\pi_a(x, y)\}$, which define the probability that the next state is $y \in \mathbb{X}$ given that the current state is $x \in \mathbb{X}$ and action $a \in \mathcal{A}$ is taken. A policy $\mathbf{u}$ can then be formally defined as a sequence of actions $\{a(t) : t \in \mathbb{Z}^+\}$, where $a(t)$ can depend only on the history of the process $\{\Phi(0), \Phi(1), \ldots, \Phi(t)\}$ and $\mathbb{Z}^+$ is the set of non-negative integers. A Markovian policy is of the form $\mathbf{u} = (u^0(\Phi(0)), u^1(\Phi(1)), u^2(\Phi(2)), \ldots)$ where u^i, for each i, is a mapping from $\mathbb{X}$ to $\mathcal{A}$ and $u^i(x) \in \mathcal{A}(x)$ for each state x. A stationary policy is then a Markovian policy with $u^i = u$ for all i and for some fixed u. Given a one-step cost function $c(\Phi(t), u(\Phi(t)))$ associated with taking action $u(\Phi(t))$ in state $\Phi(t)$, the goal of the MDP may be to find a stationary policy $\mathbf{u}$ which minimizes the objective of interest.

Variants of MDP models are defined similarly. For example, semi-Markov decision processes (SMDPs) generalize MDPs by representing the evolution of the system in continuous time, with the time spent by the process in a particular state following an arbitrary probability distribution. The decision-maker is allowed or required to choose actions whenever the system state changes (Puterman, 1994). Additional examples include partially-observable MDPs, in which there is uncertainty about the state of the process but state information can be acquired (Monahan, 1982), and constrained MDPs, which have constraints on the cumulative costs incurred at any time (Yeow et al., 2006). See also the paper by Glasserman and Yao (1994), which discusses generalized SMDPs.

After a problem is modelled as an MDP or a variant, stochastic dynamic programming methods may be applied

to either obtain numerically-optimal solutions (Sennott, 1999) or to characterize the structure of optimal policies (Stidham and Weber, 1993). For example, one common approach to finding the optimal policy is value iteration (Chen and Meyn, 1999). The reader is referred to the books by Bertsekas (2005, 2007) for an extensive coverage of topics related to the use of dynamic programming in optimal control problems, to the book on MDPs by Puterman (1994) and to the book by Meyn (2008), which provides an in-depth treatment of various topics related to scheduling in queueing systems, and talks about MDPs in detail in Chapter 9.

Stidham and Weber (1993) state that using MDPs and dynamic programming to determine optimal policies in large problems (e.g., communication networks of realistic size) is usually intractable. However, determining the structure of optimal policies for small problems can help in the development of heuristic policies for large systems. Similarly, Glasserman and Yao (1994) state that computation of optimal controls for MDPs is generally infeasible without special structure, which motivates investigation of the *form* of optimal policies. For example, in a *switching curve* policy, a threshold function is defined in terms of the states of the system: one action is optimal in the states below the threshold curve, while another is optimal for the states above the curve (Glasserman and Yao, 1994).

For the two-station problem of Figure 1 with generally-distributed inter-arrival and processing times, the state at time t is $\Phi(t) = (q_0(t), q_1(t), q_2(t))$. The actions are: to process class 0 or to process class 1 at station 1, or to idle station 1. Assuming machine 2 is non-idling, from the queueing perspective there are no scheduling decisions to be made for class 2, since it is the only class processed by machine 2; from the scheduling perspective, there is of course the question of sequencing jobs within this class. The approach of Chen et al. (1994) for this problem, which models arrival and service processes as counting processes with controllable stochastic intensities, reduces to a discrete-parameter Markov decision model (Stidham and Weber, 1993). Value iteration is then used as part of their proof of the existence of a stationary policy that is optimal for determining the actions at machine 1. This policy switches machine 1 from processing class 1 to processing class 0 (if there are jobs available in class 0) or to idling (if there are none) when the congestion level at machine 2 exceeds a threshold that is a function of the state at machine 1. Thus, the policy has a switching-curve structure, and reflects the intuition that as the congestion at station 2 grows, it becomes less and less appealing to process class 1 jobs (Chen et al., 1994). The work of Hariharan et al. (1996), which uses a discrete-time MDP model to study the generalization of our example to multiple machines, implies that when the processing times are all constant and equal, the optimal policy is to process class 0 whenever there are any class 0 jobs present and the number of jobs at station 2 is at least 2.

Moustafa (1987) shows the same result but with the assumption that exogenous arrivals are allowed at station 2. Moustafa (1996) uses a semi-Markov decision model to solve the same problem in the case of exponential inter-arrival and processing times and a cost for switching the machine between class 0 and 1 jobs.

There is a fundamental difficulty with applying the MDP approach even in the context of scheduling classes of jobs. Specifically, when buffers are infinite, the corresponding optimization problem is infinite-dimensional; when the buffers are finite, the complexity of the problem grows exponentially with the state space dimension (Meyn, 2001). Nevertheless, recent advances in the use of approximate dynamic programming (ADP) (Powell, 2010, 2012) have helped to overcome some of these computational challenges. The ADP approach is based on approximating the dynamic programming optimal cost-to-go function within the span of a given set of basis functions (Desai et al., 2012). For instance, in the context of the criss-cross network of Figure 1, Desai et al. (2012) use a smoothed approximate linear program with four basis functions: the constant function and the squared queue length for each of the three classes. Other examples of the use of approximate dynamic programming for scheduling in queueing networks include the papers by de Farias and Van Roy (2003), de Farias and Weber (2008), Moallemi et al. (2008), Veatch and Walker (2008), Veatch (2009, 2010) and Abbasi-Yadkori et al. (2014).

As mentioned above, an MDP formulation of the problem of deciding how to allocate resources to job classes can lead to both theoretical and numerical results. These results are useful in a variety of applications, e.g., in healthcare (Patrick et al., 2008; Zonderland et al., 2010). In applications where the processing time of a job can be well-estimated upon arrival or where operational-level decisions need to be made (Zonderland et al. (2010) mention the development of operational-level policies as future work), it is interesting to determine whether combining MDP results with detailed within-class scheduling methods or with the predictive-reactive framework of Bidot et al. (2009) is useful.

The rest of the queueing-theoretic approaches for scheduling that we consider are based either on models with a specific structure, or models which approximate or aggregate system characteristics.

4.2 Models with Specific Structure

One approach to addressing the complexity of scheduling problems is to restrict the policy types that are considered. In this section, we discuss four major classes of models that adopt this approach: priority queues, polling systems, vacation models and bandit models.

4.2.1 Priority Queues

The simplest models in queueing theory assume a *FCFS* discipline: jobs are processed in the order in which they arrive. A priority queueing model is different from a regular *FCFS* queue in two respects. Firstly, in a priority queueing model, arriving jobs are divided into classes or types of different priority. This notion of priority is analogous to that of job weights in scheduling: both are a measure of a job's importance. Since queueing models do not, in general, distinguish jobs based on individual characteristics, it is natural for these models to group jobs into priority classes; in scheduling, each job may be assigned a unique weight. Secondly, the system operates according to a "priority discipline", which is similar to a dispatching rule. In fact, they are based on the same idea: at a particular decision time point, be it the completion time or the arrival time of a job, one assigns an index to all jobs that are waiting to be processed and chooses the job according to the index (largest or smallest, depending on the ordering chosen) as the one to be processed next.

The queueing literature, just like the scheduling literature on dispatching rules, makes a distinction between preemptive and non-preemptive rules, and static and dynamic rules (Jaiswal, 1968).[11] For a system in which processing times become known upon arrival, an example of a priority queueing model with a static discipline is one in which a job class with index p is composed of all jobs that require between p and $p+\Delta p$ units of processing time (Adan and Resing, 2002), and jobs are sequenced in non-decreasing order of their priority index, which does not change throughout each job's time in the system. For multi-class systems in which each job class k has weight/cost c_k and an expected processing time of $\frac{1}{\mu_k}$, a classic discipline is the $c\mu$ rule (equivalently, weighted-shortest-expected-processing-time rule). This rule schedules classes in non-increasing order of their $c_k\mu_k$ values, and uses *FCFS* within each class. Jaiswal (1982) provides several examples of dynamic priority rules. In one of these, the instantaneous priority index of a job in class k is $pr_k(t) = c_k - w_k(t)$, where $w_k(t)$ is the amount of time a job has been waiting in the system up to time t. More generally, $pr_k(t)$ can be a concave function of the waiting time.

In scheduling, a dispatching rule is generally viewed as a heuristic approach that can be applied to a variety of scheduling problems, regardless of whether they are deterministic or stochastic, static or dynamic. In the literature, the performance of dispatching rules is usually evaluated experimentally, although in some cases a dispatching rule can be shown to be optimal. In addition, there is significant interest in finding rules with tight worst case bounds on performance and polynomial running time, referred to as polynomial-time approximation

[11] Jaiswal (1968) actually uses the terms *exogenous* and *endogenous* instead of *static* and *dynamic*, respectively. Nonetheless, the terms *static* and *dynamic* are also used in queueing. See, for example, the paper by Goldberg (1977).

schemes (PTAS) (Schuurman and Woeginger, 1999). An example of such a dispatching rule is the list scheduling heuristic proposed by Graham et al. (1979), in which, at a time point when a machine completes processing, the first available job from a specified priority list is scheduled. Following Schuurman and Woeginger (1999), we refer the reader to the papers by Hall (1997) and Lenstra and Shmoys (1995) for overviews of PTAS.

In queueing theory, the focus has been on theoretical performance evaluation of queueing systems operating under a particular queueing discipline and particular assumptions regarding the inter-arrival and processing time distributions. For example, consider a single-class $M/G/1$ system with arrival rate λ, mean processing time $\mathbf{E}[S]$ and system load $\rho = \lambda\mathbf{E}[S]$ ($0 \leq \rho < 1$). Let $f(\cdot)$ denote the probability density function of the processing time distribution. It has been shown that the distribution of the number of jobs in the system is the same for all non-preemptive scheduling disciplines that do not use job size information (Conway et al., 1967). As a result, the expected response time (flow time or sojourn time) $\mathbf{E}[T]$, defined as the length of time between the arrival of a job and its completion, is the same for all such policies, including *RSS*, *LCFS* and *FCFS*:

$$\mathbf{E}[T^{RSS}] = \mathbf{E}[T^{LCFS}] = \mathbf{E}[T^{FCFS}] = \mathbf{E}[S] + \frac{\lambda\mathbf{E}[S^2]}{2(1-\rho)}. \tag{1}$$

Equation (1) is based on the famous Pollaczek-Khintchine formula for the expected number of jobs in the system (Pollaczek, 1932; Khintchine, 1932). In contrast, the distribution of the response time is different for these scheduling disciplines and, in particular, $\mathbf{var}(T^{FCFS}) < \mathbf{var}(T^{RSS}) < \mathbf{var}(T^{LCFS})$ (Conway et al., 1967). The expected response time in the same system operating under the *SPT* policy, a common rule that does use processing time information, is:

$$\mathbf{E}[T^{SPT}] = \mathbf{E}[S] + \int_0^\infty \mathbf{E}[W^{SPT}(s)]f(s)ds, \tag{2}$$

where $\mathbf{E}[W^{SPT}(s)] = [\lambda\mathbf{E}[S^2]]/[2(1-\rho(s))^2]$ is the time a job of size s waits before its processing is started, and $\rho(s) = \lambda\int_0^s tf(t)dt$ is the system load consisting of jobs of size less than s only (Phipps Jr., 1956; Harchol-Balter, 2011).

Similar types of results can be obtained for multi-class systems. For instance, consider a multi-class $M/G/1$ queue with π_k being the probability that an arriving job belongs to class k. Lower-numbered classes have higher priorities (so, class 0 has the highest priority and class $|\mathcal{K}|-1$ has the lowest), and preemptions are not allowed. The random variable S_k denotes the processing time of a class k job and has distribution G_k. Define S as the "overall" processing time, drawn from distribution $G = \sum_{k=0}^{|\mathcal{K}|-1}\pi_k G_k$, and $\rho_k = \lambda\pi_k\mathbf{E}[S_k]$ as the system load

corresponding to class k jobs only. The expected delay of class k jobs in this system has been shown to be $[\lambda \mathbf{E}(S^2)]/[2(1-\sum_{k<l}\rho_k)(1-\sum_{k\leq l}\rho_k)]$ (Wolff, 1989).

Recently, there has been a strong interest in the performance evaluation of a class of policies that aims to prioritize shorter jobs in order to ensure "SMAll Response Times"; this class of "SMART" policies was introduced by Wierman et al. (2005). In their paper, Wierman et al. (2005) formalize the commonly used heuristic of giving priority to jobs which are short initially or have small remaining processing times; they derive simple bounds on the mean response time of *any* policy in the SMART class and tight bounds on the mean response time specifically for the preemptive-shortest-job-first and shortest-remaining-processing-time-first policies. Nuyens et al. (2008) further evaluate the performance of this policy class with respect to the tail of the processing time distribution.

In some cases, it can be shown that a queueing discipline is optimal under particular distributional assumptions. One of the most general results for a multi-class single-machine system is the optimality of the $c\mu$ rule in the class of *all* scheduling rules (not just priority policies) in the $M/GI/1$ queue with the possibility of idling and machine breakdowns (Meilijson and Yechiali, 1977). For our example problem, the paper by Chen et al. (1994) shows that when $c_0\mu_0 \leq (c_1 - c_2)\mu_1$, it is optimal for class 1 (type B jobs at machine 1) to have preemptive priority over class 0 (type A jobs at machine 1).

Prescriptive models that determine the best service discipline have been developed in the queueing control literature (Crabill et al., 1977). For example, Robinson (1978) uses semi-Markov decision theory to determine the optimal priority policy for deciding which of two job types a machine should process; Jaiswal (1968) describes a dynamic programming approach developed by Oliver and Pestalozzi (1965) to optimize processing time thresholds on which priorities are based, under the assumption that processing times become known upon arrival. Hassin et al. (2009) address optimization of relative priorities using an achievable region approach, which is discussed in Section 4.3.1.2. Related work includes optimization of the service discipline in a setting where there is uncertainty in the input data (Pardo and de la Fuente, 2007) and determination of the optimal thresholds for switching from a preemptive to a non-preemptive discipline (Drekic and Stanford, 2000). A general class of priority policies called fluctuation smoothing policies is proposed by Lu et al. (1994). Queueing models with switchover, which deal with the control of the service process and the queue discipline in the presence of state-dependent switching costs, are reviewed by Rosa-Hatko and Gunn (1997).

Since the underlying idea of dispatching rules and priority queueing models is the same, integration of methodologies from these two fields may be valuable. Priority queueing models can be beneficial from the perspective of

scheduling since it may be possible to cast a dispatching rule as a priority queueing discipline and apply queueing analysis in order to derive theoretical performance guarantees for this rule under particular assumptions regarding the processing or inter-arrival time distributions. As mentioned previously, there has been a significant amount of work in this direction (Harchol-Balter, 2011; Nuyens et al., 2008; Wierman et al., 2005). Conversely, new priority queueing models can be developed based on dispatching rules that have been studied in scheduling but not in queueing theory. One future work direction is to determine whether the steady state performance of composite dispatch rules, such as the Apparent Tardiness Cost heuristic (Pinedo, 2009), could be analyzed using queueing theory.

In addition, it has been noted that *any* scheduling algorithm can be represented as a function of DM, the decision mode; PF, a priority function; and AR, an arbitration rule (Jaiswal, 1982; Ruschitzka and Fabry, 1977). DM defines the time points at which the priority function PF is evaluated for all jobs in the system. The job with the highest priority value from PF is chosen for processing. AR is used to break ties between jobs with equal priorities. Representing scheduling approaches as priority policies in this manner may provide insight into the properties of these approaches, and serve as one possible framework for integrating queueing and scheduling.

4.2.2 Polling Systems

Another category of queueing-based models is known as "polling systems" (Takagi, 1988; Vishnevskii and Semenova, 2006; Boon et al., 2011). In a typical polling system, a single server has to visit and serve several queues of jobs in some order. Jobs belonging to different queues may vary in some of their characteristics, such as their inter-arrival time distributions. The server switching between the queues is equivalent to a machine or a set of machines being switched between processing of different job types. The polling system can be controlled by deciding: (i) the order in which the different queues are served (polling order); (ii) the jobs processed on each visit to a queue (queue service discipline); and (iii) the sequencing of jobs within each queue (queue service order) (Takagi, 1988; Wierman et al., 2007). Together, these decisions prescribe the order in which the jobs should be processed and, thus, form a global scheduling policy.

Polling occurs in a variety of real-world applications (Boon et al., 2011), such as flexible manufacturing (Sharafali et al., 2004). For problems that do not naturally possess a polling structure, using a polling model may still be useful. For example, a frequent assumption in polling models is the presence of switching times/costs. Therefore, such models are relevant for single machine scheduling problems with setup times/costs and, as noted by Wierman et al. (2007), for the stochastic economic lot scheduling problem. More generally, we can think of the

polling structure, together with decisions (i)–(iii), as a particular scheduling policy type, which can, theoretically, be applied to a wide range of scheduling problems. In our example problem, which does not require a polling structure, we can nonetheless apply a polling-type scheduling policy: (i) "visit" the queues corresponding to job types A and B in a cyclic order (A, B, A, etc.); (ii) during each visit, process jobs of the corresponding type until the queue is empty (exhaustive queue discipline); (iii) sequence jobs within each queue in *FCFS* order. We can then gain an understanding of the performance of this policy by using descriptive results for polling models.

Consider a polling system with $|\mathcal{K}|$ queues where for each queue k, job arrivals follow a Poisson process with rate λ_k, and processing times are distributed according to a distribution G_k with first moment γ_k and second moment $\gamma_k^{(2)}$. The load of queue k is $\rho_k = \lambda_k \gamma_k$, and the total system load is $\rho = \sum_{k=0}^{|\mathcal{K}|-1} \rho_k$. The server visits the queues in a cyclic order. If no switching time is incurred when the server switches between different queues, then this system is equivalent to an $M/G/1$ queue with arrival rate $\Lambda = \sum_{k=0}^{|\mathcal{K}|-1} \lambda_k$ and processing time distribution $\sum_{k=0}^{|\mathcal{K}|-1} (\lambda_k/\Lambda) G_k$. Since no work is created or destroyed in such a system, it has to obey a conservation law (Yechiali, 1993). Thus, if $E[W_k]$ denotes the mean waiting time for type k jobs then, regardless of the queueing discipline, the expected amount of work in the system is (Schrage, 1970; Boxma and Groenendijk, 1987):

$$\sum_{k=0}^{|\mathcal{K}|-1} \rho_k E[W_k] = \rho \frac{\sum_{k=0}^{|\mathcal{K}|-1} \lambda_k \gamma_k^{(2)}}{2(1-\rho)}. \tag{3}$$

With non-zero switching times, the above conservation law does not hold, because during the switch-over periods the server is idle while there may be work present in the system. However, useful pseudo-conservation laws (which do depend on the service discipline) have been developed. Suppose the switching times from queue k to queue $(k+1) \mod |\mathcal{K}|$ are independent and identically distributed random variables with first moment ς_k and second moment $\varsigma_k^{(2)}$. Given that the polling order is cyclic, the first and second moments of the total switching time during one cycle (length of time between visits to the same queue) are $\varsigma = \sum_{k=0}^{|\mathcal{K}|-1} \varsigma_k$ and $\varsigma^{(2)} = \sum_{k=0}^{|\mathcal{K}|-1} [\varsigma_k^{(2)} - \varsigma_k^2] + \varsigma^2$, respectively. For a system with the cyclic polling order and the exhaustive queue discipline, it has been shown that (Boxma and Groenendijk, 1987):

$$\sum_{k=0}^{|\mathcal{K}|-1} \rho_k E[W_k] = \rho \frac{\sum_{k=0}^{|\mathcal{K}|-1} \lambda_k \gamma_k^{(2)}}{2(1-\rho)} + \rho \frac{\varsigma^{(2)}}{2\varsigma} + \frac{\varsigma}{2(1-\rho)} [\rho^2 - \sum_{k=0}^{|\mathcal{K}|-1} \rho_k^2]. \tag{4}$$

The first term in this expression corresponds to the total amount of work in the system without switch-over times, while the second and third terms represent the amount of work present at an arbitrary epoch during a switch-over period (Levy and Sidi, 1990).

There has been an extensive amount of descriptive work on polling systems under particular assumptions regarding decisions (i)–(iii). We refer the reader to the work of Takagi (1986), Takagi (1988) and Levy and Sidi (1990) for overviews of such work. Below, we focus on optimizing the three different types of decisions that need to be made in polling models.

(i) Polling Order Similarly to dispatching rules, the polling order can be static or dynamic. For example, the cyclic policy, which states that the server should visit the $|\mathcal{K}|$ queues in order $0, 1, 2, \ldots, |\mathcal{K}| - 1, 0, 1, 2, \ldots$, is a static order since it is chosen prior to system operation and is independent of the system state (Levy and Sidi, 1990). An example of a dynamic polling order is one in which the server is assigned to the queue that contains the greatest amount of work at a decision epoch. Optimization of a static polling order amounts to optimizing a pattern which repeats itself every ϖ visits (referred to as a *polling table*) and has the form $Id(1), Id(2), \ldots, Id(\varpi), Id(1), Id(2), \ldots$, with $Id(i)$ being the identity of the queue in the ith position of the pattern, $0 \leq Id(i) \leq |\mathcal{K}| - 1$ (Levy and Sidi, 1990). Boxma et al. (1989; 1991; 1993) employ a three-step heuristic approach for this problem with switching times:

Step 1: Determine the relative visit frequencies fr_k for queues $k = 0, 1, 2, \ldots, |\mathcal{K}| - 1$,

Step 2: Determine the size of the polling table, ϖ, and the number of visits, num_k, that should be made to queue k, $k = 0, 1, \ldots, |\mathcal{K}| - 1$, within a cycle (note that $\varpi = \sum_{k=0}^{|\mathcal{K}|-1} num_k$),

Step 3: Given the values of ϖ, num_k and fr_k for all k, determine the order of visits to the queues within a cycle.

Each of the three steps is itself an optimization problem. For step 1, Boxma et al. (1989) propose to use the optimal proportions obtained from solving a related non-linear optimization problem in a Markovian polling system. Boxma et al. (1993) also address step 1, stating that this is the most important of the three steps. For step 2, the table size ϖ should be chosen in such a way that ensures that $\varpi fr_0, \varpi fr_1, \varpi fr_2, \ldots, \varpi fr_{|\mathcal{K}|-1}$ are integers, or within ϵ of an integer, and such that the sum of these integers equals ϖ (Boxma et al., 1993). For step 3, the goal is to find an order in which, for every k, the number of visits to other queues between two visits to queue k is approximately the same; the authors use the Golden Ratio policy proposed by Hofri and Rosberg (1987). If we fix the table size, ϖ, steps 2 and 3 become closely related to the notion of fair sequences that has received attention in the scheduling literature (e.g., see the book chapter by Kubiak (2004)): given "desired" frequencies fr_k, we may optimize some function of the differences between the actual number of visits, num_k, and fr_k so as to achieve a "fair" polling order.

Now, suppose that the system state, $(q_0, q_1, \ldots, q_{|\mathcal{K}|-1})$, where q_k is the number of jobs currently present in queue k, can be observed at the beginning of each cycle. Browne and Yechiali (1989a; 1989b) show that visiting the queues in increasing order of q_k/λ_k minimizes the expected length of the cycle. This result holds for the two most common queue service disciplines and is independent of the processing time requirements of the queues. Therefore, we can combine this rule with policies other than *FCFS* to solve more complex scheduling problems. For example, a manufacturing facility may want to minimize the length of a production cycle but also minimize the total job tardiness. In this case, we can use the above result to construct the production cycle with the minimum expected length; if we can observe processing times and due dates upon the server's arrival to a queue, then a static, deterministic scheduling problem can be solved to optimize the total tardiness of the current set of jobs. To our knowledge, such an approach has not been investigated in the literature.

The reader is referred to the papers by Levy and Sidi (1990) and Yechiali (1993) for overviews of approaches to optimization of both static and dynamic polling orders, and to the papers by Altman and Yechiali (1993), Yechiali (1991), Borst et al. (1994) and Gaujal et al. (2007) for additional examples. Khamisy et al. (1992) address the equivalent problem of positioning the queues on the cyclic path of the server (referred to as optimization of the network topology) in a polling system with precedence constraints. Bertsimas and Xu (1993) derive cost lower bounds for both static and dynamic polling orders, noting that optimization of the polling order can be seen as a vehicle routing problem in a dynamic and stochastic environment.

(ii) Queue Service Discipline The rule that is used to decide the number of jobs processed during a visit to a particular queue is the queue service discipline.[12] Two common queue service disciplines are the exhaustive and the gated disciplines. Under the exhaustive discipline, a queue is served until it becomes empty. Under the gated service discipline, all jobs that are present in the queue at the start of the visit are processed (all jobs that arrive during service have to be processed in the next visit). The exhaustive policy tends to optimize the system's efficiency, while the gated policy is known to be fairer in its allocation of the processing resource among different queues (Levy and Sidi, 1990; Levy, 1991).

The general idea for controlling the system via the queue service discipline is to employ disciplines with controllable parameters that limit the amount of service each queue receives. One option is to employ a deterministic limited policy, which has a parameter L_k that represents the maximum number of jobs that should be processed

[12]Note the difference between the earlier term "queue discipline" and the term "queue service discipline" used here.

during a visit to queue k.[13] Since even performance evaluation of this policy is difficult, there has been work on a related alternative: the binomial-gated policy (Levy, 1991). In such a policy, a parameter, $prop_k$, represents the proportion of jobs present in queue k at the start of the cycle that should be processed during this cycle. Assuming that q_k is the number of jobs in queue k when the cycle begins, the number of jobs processed is a binomial random variable with parameters q_k and $prop_k$; thus, on average, a fraction $prop_k$ of the jobs present in queue k at the beginning of the cycle will be processed during the cycle. Optimization of the system therefore amounts to finding the optimal $prop_k$ values. Since one of the main reasons this policy is considered is its analytical tractability, there is some doubt regarding its applicability in practice.

Another way to control the service discipline is to use a Bernoulli limited policy: upon completion of a job in queue k, the probability that the server will process a job from the same queue is given by a parameter ψ_k. Blanc and van der Mei (1995) study the problem of finding ψ_k values for all queues with the objective of minimizing the sum of steady-state mean waiting times weighted by arbitrary strictly positive values.

van Wijk et al. (2012) propose an approach to find service disciplines that balance efficiency and fairness. Their measure of efficiency is $\sum_{k=0}^{|\mathcal{K}|-1} \rho_k \mathbf{E}[W_k]$. Fairness is expressed in terms of the differences in mean waiting times between the queues, $\max_{k,l}(\mathbf{E}[W_k] - \mathbf{E}[W_l])$. Their goal is to optimize a weighted combination of these objectives, $\max_{k,l}(\mathbf{E}[W_k] - \mathbf{E}[W_l]) + \alpha \sum_{k=0}^{|\mathcal{K}|-1} \rho_k \mathbf{E}[W_k]$, for some $\alpha \in [0, \infty)$, by choosing the values κ_k for every queue k and implementing the corresponding κ-gated service discipline. This discipline allocates at most κ_k gated service phases to each queue k once the server arrives there. In the first phase at queue k, the server processes all jobs present at the time of its arrival to this queue. If there are jobs present in queue k once phase 1 is finished, the server stays at this queue and processes all jobs that have arrived since the start of the first phase (this corresponds to phase 2). If there are jobs in queue k when phase 2 is completed, phase 3 is started, etc., until at most κ_k phases are completed. At that point, the server switches to the next queue l and serves it using at most κ_l phases. This multi-phase approach is very similar to the notion of periodic scheduling (Bidot et al., 2009; Ouelhadj and Petrovic, 2009; Davenport and Beck, 2000), since in both cases the system is periodically reviewed and only those jobs that have arrived by the review time point are considered for scheduling.

(iii) Queue Service Order Work on scheduling within a given queue in polling systems has received little attention compared to optimization of the polling order and the queue service discipline. Fournier and Rosberg (1991) consider systems with various priority disciplines within each queue. Wierman et al. (2007) demonstrate

[13] See the paper by Levy and Sidi (1990) for a summary of variations of the limited policy.

that the order of service within the queue can have a significant impact on the performance of the system. They analyze policies that use processing time information, such as *SPT*, as well as those that do not. For instance, they consider a symmetric two-queue polling system with processing times and setup times being exponentially distributed with mean 1, and with a cyclic polling order and gated service discipline. They experimentally show that the mean waiting time under *SPT* is, on average over a variety of loads, 15% lower than that of *FCFS*. For the same system with a more variable Weibull distribution, the improvement of *SPT* over *FCFS* is even greater (Wierman et al., 2007). Boon and Adan (2009) study a polling system in which jobs within each queue are divided into low- and high-priority classes.

Optimization of the queue service order is a promising area for scheduling techniques. When a gated or a limited-type queue service discipline is employed, one needs to solve a single-machine static scheduling problem at the beginning of each queue visit. If processing times become known upon arrival, deterministic single-machine approaches may be applied; if only expected processing times are known, then a stochastic scheduling problem needs to be solved. These problems may be of varying complexity: minimizing flow time is polynomial (employing *SPT* or expected *SPT* policies is optimal) while minimizing total tardiness is in general NP-complete (Baker and Trietsch, 2009) and therefore would require a more sophisticated optimization method.

Although most of the polling literature focuses on single-server and single-station models, there have been extensions of these models to multiple-server systems (Borst, 1995; Down, 1998; Antunes et al., 2011) and networks of polling systems (Reiman and Wein, 1999; Beekhuizen et al., 2008). Scheduling decisions in such systems are the same as in single-station, single-server ones (the order of visiting the queues, the set of jobs processed on each visit and the sequence in which the jobs are to be processed), but they have to be made for every server and every station, and are dependent on the locations of all servers and the existence of constraints on the number of servers at a station at any time point, making the overall problem more complex. As a consequence, there has been very little work on optimization of scheduling decisions in such systems. An important example is the work by Browne and Weiss (1992), who consider a polling system in which the server is composed of N parallel machines which switch between queues as one unit. They address optimization of the polling order at the start of each cycle with the goal of minimizing the expected cycle length (as do Browne and Yechiali (1989a; 1989b) for a single-server system). Terekhov et al. (2012a; 2012b) and Terekhov (2013) study a polling system in which the server is composed of two machines in tandem. They investigate the performance of periodic scheduling methods

in which a static deterministic scheduling problem is solved at the beginning of each queue visit under cyclic, gated assumptions.

In summary, there are numerous avenues for integration of work in the polling model and scheduling literatures. Firstly, applying a polling-type scheduling policy to a problem implies that a vast number of descriptive results become directly applicable. Such results may be used as bounds on optimal schedules or within scheduling algorithms. Secondly, the division of the scheduling policy into three levels offers a natural framework for modelling systems with different objectives at different decision-making levels. Scheduling approaches could also prove useful for improving the performance of polling systems. Specifically, if we make the assumption that processing times of jobs become known upon arrival, then the problem of optimizing the sequence in which jobs are processed in each queue can be turned into a static, deterministic scheduling problem, making an abundance of scheduling work directly relevant. If the total workload within each queue can be observed at the beginning of each cycle, then the problem of optimizing the polling order becomes that of optimizing a system with N (batch) jobs. It would be interesting to determine whether work on lot-sizing or batch scheduling methods would be applicable in this context. The reader is referred to the work by Winands (2007) to see the connection between lot-sizing and polling systems, and is encouraged to contrast the idea of using a batch scheduling model for the problem of optimizing the polling order with the work on batch polling systems (Van Der Wal and Yechiali, 2003; Boxma et al., 2008). Additionally, future work should consider the use of polling models with precedence constraints (Khamisy et al., 1992) for modelling scheduling problems with precedences and the task management problem described by Myers et al. (2007).

4.2.3 Vacation Models

In a queueing system with vacations, the server takes "vacations" from serving a particular queue of customers. Vacations can correspond to breakdowns of a machine, maintenance operations, or processing of other job classes (Stidham, 2002; Doshi, 1986). Consider the example of Figure 1. From the perspective of a class 0 job, the time spent processing class 1 can be viewed as a machine vacation. Adopting this point of view, one can see that determining how to allocate machine capacity among job classes is equivalent to determining the timing and duration of vacations. In fact, a vacation model can be seen as a special case of a polling system. Thus, we do not review vacation models in as much detail as the work on polling systems. We note, however, that while the methodology employed in the study of vacation models is similar to that of polling systems, some more general results have been obtained (Stidham, 2002). One of the most important results is that the waiting time

in an $M/GI/1$ queue with vacations follows the distribution of the sum of two independent components: the waiting time in an $M/GI/1$ queue without vacations and the equilibrium residual vacation time (Stidham, 2002; Fuhrmann, 1984).

Vacation models could be useful for scheduling in the same way as work on optimization of the polling order and queue service discipline in polling systems. In particular, vacation models can provide high-level guidance regarding how much time should be spent on a specific class prior to taking a vacation. A review of descriptive vacation models can be found in the paper by Doshi (1986), while various prescriptive vacation models are discussed by Tadj and Choudhury (2005). Extensions of vacation models to the case of multiple servers have been analyzed by, for example, Kao and Narayanan (1991) and Chao and Zhao (1998). The book by Tian and Zhang (2006) provides a comprehensive review of both descriptive and prescriptive vacation models.

4.2.4 Bandit Models

A multi-armed bandit (MAB) problem (Gittins, 1979; Whittle, 1988; Bertsimas, 1995; Bertsimas and Niño-Mora, 1996) consists of a set $\mathcal{N}$ of jobs, only one of which can be processed at each discrete point in time. For every time point t when a job j in state $x_j(t)$ is being worked on, a reward $R^j_{x_j(t)}$ is obtained. The rewards are assumed to be additive and are discounted in time by a factor φ, $0 < \varphi < 1$. The job that is being executed changes state according to a homogeneous Markov transition rule, while the states of the jobs that are not being worked on do not change. The goal of the problem is to determine a scheduling policy: a rule that, at each point in time, prescribes which job should be executed and maximizes the total expected discounted reward over an infinite horizon (Bertsimas and Niño-Mora, 1996). Interestingly, in computer science, significant attention (Streeter, 2007; Gagliolo and Schmidhuber, 2007; Cicirello and Smith, 2005; Radlinski et al., 2005; Vermorel and Mohri, 2005) has been given to a variation of the MAB problem in which the distribution of rewards is not known in advance and the goal is to choose, at each iteration, the job to execute in order to maximize the sum of collected rewards.

The initial inspiration for the study of MABs is the problem faced by a gambler in a casino deciding which slot machine should be played (Puterman, 1994) or, equivalently, which lever or arm of a slot machine should be pulled (Weber and Weiss, 1990). Naturally, the MAB problem is also a representation of the situation where a decision-maker needs to choose which job to execute at a point in time, with states defined by the jobs' levels of completion and a reward being obtained only when a job is finished (Puterman, 1994). From the perspective of scheduling, the MAB problem is a representation of a single machine scheduling problem with preemptions in

which the set of jobs stays fixed over time (Pinedo, 2009). The state of the job can be the remaining processing time, which changes only when this job is the one being processed by the machine. Bertsimas and Niño-Mora (1996) state that the MAB problem is a special case of a dynamic and stochastic job scheduling problem. An additional application of MABs is in sequential clinical trials where a new medical treatment is compared to an existing one or to a placebo (Puterman, 1994).

The arm-acquiring bandit problem is an extension of the MAB problem in which a set of new jobs arrives at time t. These jobs can be executed starting at time $t + 1$ and are assumed to be independent of each other and of all the previous jobs. As in the MAB problem, the goal is to determine a policy that specifies the job that should be executed at each point in time (Mahajan and Teneketzis, 2007). This problem can be used as a representation of a single machine dynamic scheduling problem with preemptions. The example of Figure 1 can be viewed as an arm-acquiring bandit problem if we allow preemption. Within the queueing literature, the MAB problem is related to polling models, since the decision to execute a particular job can be equivalent to choosing a particular queue.

There are many other variations of MAB models, most of which can be linked to problems from the scheduling literature. These include:

- the MAB problem with switching penalties (Mahajan and Teneketzis, 2007), corresponding to a static single machine preemptive scheduling problem with switching costs.

- the MAB problem with deadlines (Niño-Mora, 2007), corresponding to a static single machine preemptive scheduling problem with deadlines.

- the branching bandit problem (Varaiya et al., 1985; Weiss, 1988), in which jobs are classified into types according to their state and where a job is replaced by some number of new jobs of each type upon its completion. This problem allows modelling of job arrival processes that are more general than Poisson and can be used to represent a machine in a job shop that processes a variety of parts (Bertsimas et al., 1995).

- the restless bandit problem (Weber and Weiss, 1990; Niño-Mora, 2007), in which ϱ jobs have to be executed at any time point; rewards may be incurred and states may change even for jobs that are not being worked on. In the literature, it is stated that these models can represent situations where ϱ out of all available workers always need to be active, and their states represent their physical condition; states change regardless of whether the worker is busy or idle (e.g., if the workers are getting rest, their physical condition improves)

(Weber and Weiss, 1990). Similarly, we can use the restless bandit problem to model a manufacturing environment with ϱ parallel machines; this model is general enough to represent deterioration and improvement in the condition of machines. Future research should determine the relationship between the restless bandit problem and stochastic variations of resource constrained project scheduling discussed by Mercier and Van Hentenryck (2008). The restless bandit model can also be applied in the context of the system shown in Figure 1, since we can think of type A and B jobs as two jobs which change state depending on the amount of processing received and on the amount of work that arrives.

- the MAB problem with a goal state (Dumitriu et al., 2003; Katta and Sethuraman, 2005), which is related to planning problems in artificial intelligence (Ghallab et al., 2004).

One of the biggest advantages of the MAB problem representation and its variants comes from a powerful class of policies that have been developed for solving them. These policies are based on a priority *index* that is defined for each job as a function of its state. At each time point, the set of jobs with the greatest priority index values is chosen for processing (Niño-Mora, 2007).[14]

Interestingly, the earliest result on the optimality of priority index rules arose in the context of a *deterministic* problem (Niño-Mora, 2007). Specifically, Smith (1956) showed that an index rule is optimal for the problem of minimizing the sum of completion times of a set of jobs with known processing times and linear holding costs (weights). In this setting, the index of a job can be defined as the ratio of the holding cost rate per unit of time to its processing time, which represents the cost reduction per unit of effort, or the average productivity of work on the job (Niño-Mora, 2007). Subsequently, Rothkopf (1966) extended Smith's result to the problem with stochastic job durations. Cox and Smith (1961) showed the optimality of an equivalent index rule in the context of a multi-class single-server queue with linear holding costs. The seminal work by Klimov (1975) develops an optimal index rule for a more complex version of this problem, one with Bernoulli feedback between job classes (Niño-Mora, 2007).

Both the MAB problem and the arm-acquiring bandit problem are solvable by a Gittins index policy (Gittins, 1979; Bertsimas and Niño-Mora, 1996; Mahajan and Teneketzis, 2007), which is a generalization of the well-known $c\mu$-rule (also discussed in the context of priority queues above, and in the context of fluid models and the achievable region method in Section 4.3). According to such a policy, at each time point t, one should process the job with the highest value of the Gittins index, which represents the maximum expected discounted reward per

[14] These policies are of the same nature as those discussed in the priority queues section above, and they can be similarly classified into static and dynamic policies.

unit of expected discounted time due to the processing of a job (Mahajan and Teneketzis, 2007). Heuristics based on the Gittins index have also been developed for more complex problems, such as research planning (Glazebrook and Owen, 1995). For the restless bandit problem, Whittle (1988) provides an index policy based on the solution of its relaxation, while Weber and Weiss (1990) show that this policy is asymptotically optimal. The paper by Niño-Mora (2007) provides a unifying approach for developing and computing index-based priority policies, illustrating a number of application areas, including scheduling in multi-class queues and dynamic priority allocation to multiple stochastic jobs. We note that MDP methods (Section 4.1) and achievable region methods (Section 4.3) have both been used to examine bandit problems (Puterman, 1994; Bertsimas, 1995).

The simplicity of index policies, together with their theoretical optimality in some settings, implies that it may be useful to model part or all of a dynamic scheduling problem as a bandit problem. For instance, a multi-armed or an arm-acquiring bandit problem could be used as a relaxation of a non-preemptive scheduling problem that is solved at each scheduling point in a predictive-reactive method. Alternatively, the Gittins indices of the jobs could be used to create effective scheduling heuristics or to guide a predictive-reactive approach to better decisions, since these indices can be incorporated into the constraints or the objective function of the models used to construct predictive schedules.

4.3 Methods Based on Alternative Representations

In the previous section, we reviewed methods that deal with the complexity of scheduling in queueing systems by restricting the type of policies considered. In this section, we discuss a methodology that is based on developing alternative, simpler representations of the system of interest via approximations or abstractions. This methodology consists of four steps:

1. formulation of the alternative representation for the problem of interest;
2. solution of the formulation;
3. derivation of an implementable scheduling policy for the original scheduling problem from the solution to the alternative representation formulation;
4. performance analysis of the solution.

We discuss several options for steps 1 and 2 in the following section. General approaches for step 3 are discussed in Section 4.3.2. Step 4 is outside the scope of this paper, and the reader is encouraged to consult the papers

referenced throughout this section.

4.3.1 Alternative Representations

The alternative representations we present here are based on two main approaches. The first is system approximation via aggregation of time and state space, and the two specific approximations we consider are Brownian models and fluid models. The second approach, called the achievable region method, uses aggregation of a different type: since multiple policies may result in the same performance, it maps the problem of finding the optimal control to the lower-dimensional problem of finding the optimal performance vector.

4.3.1.1 Approximations While the idea of approximating the problem of interest by a simpler one has also been used in scheduling, the characteristics that are relaxed to develop the approximation are typically different than those used in queueing. In scheduling, approximation techniques are based on relaxing precedence or no-preemption constraints; in queueing, approximations are based on scaling time and space in order to view high level patterns in the system's behaviour.

Chen and Mandelbaum (1991) present the approximations used in queueing theory using a three-level aggregation framework that is akin to the well-known abstraction framework for scheduling in supply chains (see, e.g., the book by Pinedo (2009) or the paper by Kreipl and Pinedo (2004)). The lowest, microscopic level of their framework corresponds to the original discrete stochastic network of interest; this level requires a detailed representation of the jobs present in the system. At the highest, macroscopic level, the system is approximated by a deterministic fluid model that captures the system's long-run average behaviour and requires minimal data. The intermediate, mesoscopic level represents the deviations between the original network and its fluid model, modelled by stochastic diffusion approximations (Chen and Mandelbaum, 1991). These three levels correspond to the operational, tactical and strategic levels of scheduling in a supply chain. Although this traditional supply chain framework is not based on a formal scaling procedure, it also utilizes the ideas of scaling time and state space, since moving from operational-level models to strategic-level ones requires decreasing the clock speed and product differentiation (Kreipl and Pinedo, 2004).

The mesoscopic diffusion models we discuss here are Brownian models, which arise, due to a Functional Central Limit Theorem, as a limit of a sequence of systems in which utilization is increased to 1 while the number of machines is fixed and the probability of waiting increases to 1 (Chen and Mandelbaum, 1991; Halfin and Whitt, 1981). For alternative heavy traffic scaling approaches, such as when the arrival rate and the number of machines

are increased to infinity while utilization is fixed, or when utilization increases to 1 but the probability of waiting is fixed to a value below 1, we refer the reader to the paper by Halfin and Whitt (1981) and the references therein. The macroscopic fluid models we review are a result of a rescaling procedure based on a Functional Law of Large Numbers (Chen and Mandelbaum, 1991). For a review of fluid and diffusion approximations, we refer the reader to the papers by Chen and Mandelbaum (1994a; 1994b).

Brownian control problems in higher than two dimensions are, in general, hard to solve. This is due to the fact that determining the stationary distribution of higher dimension reflected Brownian motion is difficult, except in cases where there is state-space collapse (Williams, 1998) or the stationary distribution has product form (Dieker, 2010). Thus, although Brownian models capture variability better than fluid models, and the fluid model can be thought of as a coarser approximation method, fluid models are usually easier to work with. In some cases, it is necessary to establish some specific characteristic of the system via the fluid model before a Brownian model can be built (Kang et al., 2004). Interestingly, authors such as Maglaras (2003) have an alternative view on the use of these two approximations, using the Brownian model for a description of an asymptotically optimal policy at the macroscopic level, and the fluid model as a translation mechanism to derive an implementable policy. Veatch (2003) provides a comparison of diffusion and fluid models in two queueing networks.

Brownian Models In heavy-traffic conditions, when the utilization of a system approaches its capacity, costs are amplified and optimal control is even more important than in under-utilized systems (Stidham, 2002). It has been shown that the queue length or the workload process of a queueing network with balanced heavy loading[15] can be approximated by a diffusion process called a reflecting Brownian motion (Williams, 1996). The resulting Brownian models have the advantages of requiring a minimal amount of data and of being based on a compact mathematical representation (Harrison, 2003). Moreover, they can be used for networks with multiple classes of jobs, both feedforward and feedbackward routings and machines subject to various types of disruptions (Chen and Yao, 2001).

Specifically, Brownian models are constructed by compressing time by a factor of n and compressing the spatial dimensions by a factor of $\sqrt{n}$ (Harrison, 1996). More formally, Brownian approximations arise as a limit of a sequence of systems indexed by n as $n \to \infty$ (Kelly and Laws, 1993). When looking at the system on this scale, it is impossible to pay attention to detailed scheduling decisions, but *trends* in the behaviour of the

[15]For a general open network, this term refers to the situation when the load imposed on each station by some exogenous input process is approximately equal to the capacity of that station. In a closed queueing network, the term implies that the total population within the network is large and the relative intensities for the different stations are approximately the same (Harrison, 1988).

system become apparent. Therefore, solving the Brownian scheduling problem amounts to determining high level properties such as conditions for idling the machine(s). These properties can then be translated into implementable scheduling policies. This approach is best illustrated using the two-station example presented at the beginning of Section 4 and the paper of Harrison and Wein (1989).

Recall that $q_k(t)$ is the total number of class k jobs present at time t. Let $i_m(t)$ be the total amount of time that machine m is idle in $[0,t]$. The scaled versions of these quantities, denoted by a tilde, are:

$$\tilde{q}_k(t) = \frac{q_k(nt)}{\sqrt{n}}, \ \ t \geq 0, \text{ and } k = 0,1,2, \tag{5}$$

$$\tilde{i}_m(t) = \frac{i_m(nt)}{\sqrt{n}}, \ \ t \geq 0, \text{ and } m = 1,2. \tag{6}$$

In order to define the Brownian model in terms of workload present in the system, let M_{mk} be the expected amount of time that should be given by machine m to a class k job before it leaves the system (Harrison and Wein, 1989); denote the workload profile matrix made up of the M_{mk} values by $\mathbf{M}$. Let $\tilde{\mathcal{W}}_m = \{\tilde{\mathcal{W}}_m(t), t \geq 0\}$ be the scaled workload process where $\tilde{\mathcal{W}}_m(t) = \sum_{k=0}^{2} M_{mk}\tilde{q}_k(t)$, $t \geq 0$ and $m = 1,2$. $\tilde{\mathcal{W}}_m(t)$ is therefore the total expected amount of scaled work left for machine m at time t anywhere in the system. It has been shown by Harrison (1988) that the sequencing problem at machine 1 is well-approximated by the Brownian control problem of choosing processes $\{\tilde{q}_k(t), t \geq 0, k = 0,1,2\}$ and $\{\tilde{i}_m(t), t \geq 0, m = 1,2\}$ which are right continuous with left limits and are a solution to the following workload formulation:

$$\text{minimize } \lim_{\mathcal{T}\to\infty} \sup \qquad \frac{1}{\mathcal{T}} E\Big[\int_0^{\mathcal{T}} \sum_{k=0}^{2} \tilde{q}_k(t)dt\Big] \tag{7}$$

$$\text{subject to } \ \frac{1}{2}\tilde{q}_0(t) + \frac{1}{2}\tilde{q}_1(t) = \mathcal{B}_1(t) + \tilde{i}_1(t), \ \forall t \geq 0, \tag{8}$$

$$\tilde{q}_1(t) + \tilde{q}_2(t) = \mathcal{B}_2(t) + \tilde{i}_2(t), \ \forall t \geq 0, \tag{9}$$

$$\{\tilde{i}_m(t), t \geq 0, m = 1,2\} \text{ is non-decreasing with } \qquad \tilde{i}_m(0) = 0, m = 1,2, \tag{10}$$

$$\tilde{q}_k(t) \geq 0, \qquad \forall t \geq 0, k = 0,1,2, \tag{11}$$

$$\{\tilde{q}_k(t), t \geq 0, k = 0,1,2\} \text{ is non-anticipating} \qquad \text{with respect to } \mathbf{X}, \tag{12}$$

$$\{\tilde{i}_m(t), t \geq 0, m = 1,2\} \text{ is non-anticipating} \qquad \text{with respect to } \mathbf{X}. \tag{13}$$

In this formulation, $\mathbf{X}$ is a three-dimensional Brownian motion[16] with drift vector θ and covariance matrix $\mathbf{\Gamma}$, and $\mathcal{B}(t) = \mathbf{M}X(t)$, so that $\boldsymbol{\mathcal{B}} = (\mathcal{B}_1, \mathcal{B}_2)$ is a two-dimensional Brownian motion with drift $\mathbf{M}\theta$ and covariance

[16] See Chapter 10 of the book by Ross (2003) for an introduction to Brownian motion.

matrix $\mathbf{M\Gamma M}^T$. In fact, there exists a solution that, with probability 1, simultaneously minimizes $\sum_{k=0}^{2} \tilde{q}_k(t)$ for all t.

To obtain this solution, assume that the processes on the right-hand side of Equations (8) and (9) are known processes $b_1 = \{b_1(t), t \geq 0\}$ and $b_2 = \{b_2(t), t \geq 0\}$, respectively, where $b_m(t) = \mathcal{B}_m(t) + \tilde{i}_m(t)$, $m = 1, 2$, the process $\{\tilde{i}_m(t), t \geq 0, m = 1, 2\}$ satisfies Equations (10) and (13), and b_1 and b_2 are nonnegative (Harrison and Wein, 1989). Given these processes, it can be observed that the following linear program is embedded in the above workload formulation at time t:

$$\begin{aligned}
\text{minimize} \quad & \tilde{q}_0(t) + \tilde{q}_1(t) + \tilde{q}_2(t) && (14)\\
\text{subject to} \quad & \frac{1}{2}\tilde{q}_0(t) + \frac{1}{2}\tilde{q}_1(t) = b_1(t), && (15)\\
& \tilde{q}_2(t) + \tilde{q}_2(t) = b_2(t), && (16)\\
& \tilde{q}_k(t) \geq 0, k = 0, 1, 2. && (17)
\end{aligned}$$

This linear program can have different non-negative right-hand side values at each time t. The corresponding dual variables represent the increase in the value of objective function (14) per unit increase in the value of $b_m(t)$. Since $b_1(t)$ and $b_2(t)$ are non-negative, the solution to the dual will also be non-negative. Consequently, solving the problem stated in Equations (7)–(13) amounts to finding $\tilde{i}_1$ and $\tilde{i}_2$ that minimize b_1 and b_2, respectively, and then, for every time point, solving the linear program (14)–(17). Defining b_m^* as this minimum value for machine m, i.e., $b_m^*(t) = \mathcal{B}_m(t) - \inf_{0 \leq s \leq t} \mathcal{B}_m(s)$, the solution to the linear program is:

$$\begin{aligned}
\tilde{I}_m^*(t) &= -\inf_{0 \leq s \leq t} \mathcal{B}_m(s) \quad \text{for} \quad m = 1, 2, && (18)\\
\tilde{q}_0^*(t) &= [2b_1^*(t) - b_2^*(t)]^+, && (19)\\
\tilde{q}_1^*(t) &= [2b_1^*(t) \wedge b_2^*(t)], && (20)\\
\tilde{q}_2^*(t) &= [b_2^*(t) - 2b_1^*(t)]^+, && (21)
\end{aligned}$$

and $\sum_{k=0}^{2} \tilde{q}_k^*(t) = [2b_1^*(t) \vee b_2^*(t)]$, $t \geq 0$, where $[a]^+$ denotes $\max\{0, a\}$, $[a \wedge b]$ denotes $\min\{a, b\}$ and $[a \vee b]$ denotes $\max\{a, b\}$.

Based on the definition of $\tilde{\mathcal{W}}_m(t)$ and Equations (8), (9) and (18), this solution can be interpreted as follows: $\tilde{i}_m^*(t)$ increases only at times t when $\tilde{\mathcal{W}}_m^*(t) = 0$ for $m = 1, 2$. Therefore, in this solution, machine 2 incurs scaled idleness only when there are no scaled jobs of type B *anywhere in the network* (Harrison and Wein, 1989). The interpretation of the above solution suggests that the policy for the original network should attempt to avoid

machine 2 idleness when there are type B jobs in the system. Finding a policy that implements this notion in a way that is optimal for the original system is non-trivial. For example, one intuitive implementation is a policy that always gives priority to type B jobs at station 1. This policy, however, will tend to keep a greater total number of jobs in the system than necessary since it will delay the processing of type A jobs, which would have otherwise left the system faster. Thus, a better approach is a policy that gives priority to type A if the number of jobs at station 2 is greater than some value ψ and gives priority to type B otherwise. This policy balances the short-run objective of reducing the number of jobs in the system (due to giving priority to type A jobs some of the time) and the longer-run objective of avoiding idleness at station 2. The best value of the parameter ψ and the asymptotic behaviour of the proposed policy (Step 4 of the above framework) are discussed by Harrison and Wein (1989). The papers by Martins et al. (1996) and Kushner and Martins (1996) provide rigorous proofs of the connection between the problem of scheduling in this network and the diffusion process that is its heavy traffic limit. Additional work on scheduling in the criss-cross network based on Brownian models includes the papers by Kumar and Muthuraman (2004) and Budhiraja and Ghosh (2005).

Brownian approximation models have been applied to scheduling problems in a variety of queueing settings, including closed networks (Chevalier and Wein, 1993), networks with abandonments (Kim and Ward, 2013), networks with both controllable arrivals and queue disciplines (Wein, 1990) and networks representing a hospital emergency department (Huang, 2013). Wein (1994) uses Brownian approximations to address the bi-criteria problem of minimizing the long-run expected average of a linear combination of job sojourn time and sojourn time inequity between different job classes. For a comprehensive overview of Brownian models, the reader is referred to the book by Harrison (2013).

Fluid Models Another way to approximate the problem of scheduling in a multi-class queueing system is to relax the assumption that the system processes discrete jobs. If we consider the behaviour of the system over a long time horizon, this makes intuitive sense – viewing the evolution of the system at a high level would give one the impression of continuous fluids moving through the system, rather than discrete entities. Thus, a multiclass fluid network is different from the network it approximates because it processes continuous fluid flows rather than discrete jobs. Scheduling of this network corresponds to allocation of the available processing capacity of each station to the various fluid classes (Chen and Yao, 1993). The fluid solution can then be translated into scheduling decisions for the original system in both static and dynamic conditions.

Fluid model work on classical, deterministic scheduling problems is based on the fundamental result that the

optimal fluid makespan provides a lower bound on the optimal makespan in a high-multiplicity[17] job shop. The optimal fluid makespan is equal to the workload on the bottleneck machine (known as the *machine lower bound* in the scheduling literature) (Bertsimas and Gamarnik, 1999).[18] These results are used by Bertsimas and Gamarnik (1999), Boudoukh et al. (2001) and Bertsimas and Sethuraman (2002) to develop heuristic algorithms that produce asymptotically optimal schedules as the number of jobs of each class (the multiplicity) grows. Bertsimas et al. (2003) apply the same ideas to the harder problem of minimizing inventory holding costs in a job shop. Dai and Weiss (2002) address minimization of makespan in a more general setting: the processing times of jobs in a class are different but follow the same distribution. Nazarathy and Weiss (2009, 2010) study high-volume job shops, where the number of jobs is large in comparison to the number of machines and the maximum number of activities per job, with weighted flow time and makespan objectives, respectively.

The paper of Nazarathy and Weiss (2010), in addition, shows how a typical job shop scheduling problem can be represented as a multi-class queueing network problem and provides a step toward the same goal of integration of queueing and scheduling as this paper. Specifically, consider the job shop scheduling problem with a set $\mathcal{N}$ of jobs and a set $\mathcal{M}$ of machines. Each job j consists of a set of activities. Activity i of job j is processed[19] on machine $\sigma(i,j)$ and has duration $s_{i,j}$. In a high-volume job shop, $|\mathcal{N}|$ is large, but $|\mathcal{M}|$ is fixed and the number of activities per job is bounded. This problem can be modelled as a multi-class queueing network as follows. The activities are classified into a set $\mathcal{K}$ of classes, and to be consistent with queueing terminology, we further refer to these as jobs of class k.[20] A job of class k is processed by machine $\sigma(k)$, and the set of classes processed by machine m is denoted $\mathcal{K}_m$. Looking at the processing times of all jobs in class k leads to a processing time distribution G_k with rate μ_k. By considering the next routing step possible after the completion of any class k job, define $\pi_{k,l}$ as the fraction of class k jobs that, upon completion, turn into class l jobs, and $1-\sum_l \pi_{k,l}$ as the fraction of class k jobs that, upon completion, leave the system. In summary, the corresponding multi-class queueing network consists of $|\mathcal{M}|$ machines and $|\mathcal{K}|$ classes, with jobs of class k processed according to distribution G_k and routed according to the probability matrix $\mathbf{\Pi}$ defined by the $\pi_{k,l}$ values, further referred to as the routing matrix. The initial number of jobs in each class is $q_k(0)$, $k=0,1,\ldots,|\mathcal{K}|-1$, with $|\mathcal{N}|=\sum_{k=0}^{|\mathcal{K}|-1} q_k(0)$. There are no exogenous arrivals; hence, this system is called a *finite-horizon* multi-class queueing network (Nazarathy and Weiss, 2010).

[17] The term *high-multiplicity* refers to the fact that more than one job is present in each class.

[18] In the scheduling literature, it is known that the maximum, over all machines, of the total processing time required by the jobs provides a lower bound for the makespan of the problem.

[19] A job is allowed to visit the same machine more than once along its route.

[20] Recall that an instance of a class k job is equivalent to an *activity* in the scheduling literature.

The dynamics of the fluid model corresponding to this queueing network are based on the following equation:

$$\bar{q}_k(t) = \bar{q}_k(0) - \mu_k \int_0^t \bar{u}_k(y)dy + \sum_l \pi_{l,k}\mu_l \int_0^t \bar{u}_l(y)dy, \tag{22}$$

where $\bar{q}_k(t)$ is the amount of fluid present in class k at time t and $\bar{u}_k(y)$ is the instantaneous allocation of the processing capacity of machine $\sigma(k)$ to class k. Fluid quantities are denoted by a bar, e.g., $\bar{q}_k(t)$, for the rest of the paper. Equation (22) states that the total amount of fluid in class k at time t is equal to the initial amount of fluid minus the amount that has been processed up to time t, plus the amount that has arrived from other classes by time t. Let $\bar{q}_k^{(+)}(t)$ be the total amount of class k fluid that still needs to flow through class k (including fluid currently in other classes that will turn into class k eventually). Assuming that T_m denotes the total workload for machine m, let $T^* = \max\{T_1, \ldots, T_{|\mathcal{M}|}\}$ be the machine lower bound for the job shop. The solution to the fluid makespan minimization problem is shown by Nazarathy and Weiss (2010) to be:

$$\bar{u}_k(t) = \frac{\bar{q}_k^{(+)}(0)}{\mu_k T^*}, \tag{23}$$

$$\bar{q}_k(t) = \bar{q}_k(0)(1 - \frac{t}{T^*}), \tag{24}$$

$$\bar{q}_k^{(+)}(t) = \bar{q}_k^{(+)}(0)(1 - \frac{t}{T^*}). \tag{25}$$

Let $q_k(t)$ be the number of class k jobs that are present in the queue of machine $\sigma(k)$ at time t, and $q_k^{(+)}(t)$ be the total number of class k jobs that still need to be completed at time t (including jobs of class k that have not yet arrived at $\sigma(k)$). The jobs can then be scheduled via an online fluid tracking policy:

for each time t and each machine m that is free

define $\mathcal{K}_m(t) = \{k \in \mathcal{K}_m : q_k(t) > 0\}$,

if $\mathcal{K}_m(t) = \emptyset$

idle machine m

else

process an available job of class k^* which achieves the maximum, over the set $\{l \in \mathcal{K}_m(t)\}$, of $\frac{q_l^+(t)}{q_0^+(t)}$.

Ties for k^* and the choice of job within k^* can be decided using arbitrary rules. We refer the reader to the paper by Nazarathy and Weiss (2010) for additional details as well as an overview of other job-shop scheduling rules based on the solution of the fluid model. We also refer the reader to the thesis by Raviv (2003) which demonstrates that fluid models can be used as approximations of large instances of other hard combinatorial problems.

The fluid policy described above can be applied in a dynamic setting if the fluid model on which it is based is adjusted to include exogenous arrival information. In this case, Equation (22) becomes

$$\bar{q}_k(t) = \bar{q}_k(0) + \lambda_k(t) - \mu_k \hat{\bar{u}}_k(t) + \sum_l \pi_{l,k}\mu_l \hat{\bar{u}}_l(t), \tag{26}$$

where $\hat{\bar{u}}_k(t) = \int_0^t \bar{u}_k(y)dy$ is the cumulative amount of time that station $\sigma(k)$ devotes to class k in $[0,t]$ (Chen and Yao, 1993).

Suppose we would like to minimize holding costs in a network with $|\mathcal{K}|$ classes and $|\mathcal{M}|$ machines, $|\mathcal{M}| \leq |\mathcal{K}|$, arrival rate vector $\boldsymbol{\lambda} = (\lambda_0, \ldots, \lambda_{|\mathcal{K}|-1})^T$ and processing rate vector $\boldsymbol{\mu} = (\mu_0, \ldots, \mu_{|\mathcal{K}|-1})^T$. Assume the system's routing matrix is $\mathbf{\Pi}$ and the constituency matrix is $\mathbf{C}$. Recall that the constituency matrix specifies which classes are processed by which machines, i.e., each entry (m,k) is 1 if class k can be processed by machine m and 0 otherwise. Define $\mathbf{A} = (\mathbf{I} - \mathbf{\Pi}')diag(\mu)$, where $diag(\mu)$ is a diagonal matrix of processing rates. $\mathbf{A}$ is referred to as an *input-output matrix* in stochastic processing network terminology, since each of its entries represents the average rate at which a particular machine depletes a particular class (Teh, 2000) (see below for an example). Let $\mathbf{1}$ be a vector of 1s of the appropriate dimension, $\bar{\mathbf{q}}(t) = (\bar{q}_0(t), \ldots, \bar{q}_{|\mathcal{K}|-1}(t))^T$ and $\hat{\bar{\mathbf{u}}}(t) = (\hat{\bar{u}}_0(t), \ldots, \hat{\bar{u}}_{|\mathcal{K}|-1}(t))^T$. Recall that in the description of Brownian models, $i_m(t)$ was used to denote the total idleness in $[0,t]$; we assume the same meaning here and, using the appropriate fluid scaling, define $\bar{\mathbf{i}}(t) = (\bar{i}_1(t), \ldots, \bar{i}_{|\mathcal{M}|}(t))$. Let c_k be the per time unit holding cost for jobs of class k, and $\mathbf{c} = (c_0, \ldots, c_{|\mathcal{K}|-1})$. The fluid model for the problem of minimizing the total holding cost is:

$$\begin{aligned}
\text{minimize} \quad & \int_0^\infty \mathbf{c}\bar{\mathbf{q}}(t)dt && (27)\\
\text{subject to } \bar{\mathbf{q}}(t) &= \bar{\mathbf{q}}(0) + \boldsymbol{\lambda}t - \mathbf{A}\hat{\bar{\mathbf{u}}}(t) \ \ \forall\, t, && (28)\\
\bar{\mathbf{q}}(t) &\geq 0, \ \ \forall\, t, && (29)\\
\hat{\bar{\mathbf{u}}}(t) & \quad \text{is non-decreasing with } \hat{\bar{\mathbf{u}}}(0) = 0, \forall\, t, && (30)\\
\bar{\mathbf{i}}(t) &= \mathbf{1}t - \mathbf{C}\hat{\bar{\mathbf{u}}}(t) \text{ is non-decreasing}, \forall\, t. && (31)
\end{aligned}$$

For the example problem of Figure 1, $\bar{\mathbf{q}}(t) = (\bar{q}_0(t), \bar{q}_1(t), \bar{q}_2(t))^T$, $\boldsymbol{\lambda} = (\lambda_0, \lambda_1, 0)^T$, $\mathbf{c} = (c_0, c_1, c_2)^T$,

$$\mathbf{\Pi} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} \mu_0 & 0 & 0 \\ 0 & \mu_1 & 0 \\ 0 & -\mu_1 & \mu_2 \end{pmatrix},$$

and

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

One approach for obtaining a solution to the above fluid control problem is based on the idea that for some (possibly small) period of time, say $[0, t_1]$, the allocation of capacity remains a constant proportion, i.e., $\hat{\hat{\mathbf{u}}}(t) = \mathbf{u}t$ with a fixed $\mathbf{u}$ for all $t \in [0, t_1]$ (Chen and Yao, 1993). Each component of the column vector $\mathbf{u}$, u_k, $k = 0, \ldots, |\mathcal{K}| - 1$, represents the proportion of capacity that station $\sigma(k)$ should devote to processing class k fluids (Atkins and Chen, 1995) and forms the solution to the following linear program (Atkins and Chen, 1995; Chen and Yao, 1993):

$$\begin{aligned} \text{maximize} \quad & (\mathbf{cA})\mathbf{u} && (32) \\ \text{subject to} \quad & (\mathbf{Au} - \boldsymbol{\lambda})_k \leq 0, \quad k \in \mathcal{S}, && (33) \\ & \mathbf{Cu} \leq \mathbf{1}, && (34) \\ & \mathbf{u} \geq 0. && (35) \end{aligned}$$

The set $\mathcal{S}$ is a subset of $\{0, \ldots, |\mathcal{K}| - 1\}$ corresponding to classes with 0 fluid level. Constraint (33) ensures that the fluid level for classes in the set $\mathcal{S}$ does not go below 0, while constraint (34) states that any chosen allocation $\mathbf{u}$ has to satisfy the unary capacity of each machine. The linear program is re-solved at different decision epochs and, when the fluid level in a particular class becomes 0, that class is added to the set $\mathcal{S}$. Atkins and Chen (1995) create state-dependent priority rules based on the solution of the linear program and compare their performance to traditional policies such as *FCFS* and *SPT* in four environments. They conclude that the performance of fluid policies is at least comparable to classical scheduling heuristics regardless of the distributions of processing or inter-arrival times and the traffic intensities. They state that more research is necessary to highlight the advantages of fluid policies, which may become more apparent in non-stationary situations or when there are machine breakdowns.

In some cases, it is possible to solve the linear program (32)–(35) for all possible states of the system a priori and use the resulting information to develop an online scheduling policy. For example, consider a single station serving four classes, with c_0 being the largest of the holding costs. Solving the linear program for any state when there is fluid present in class 0 results in an optimal solution $u_0^* > 0$, $u_1^* = u_2^* = u_3^* = 0$. This solution can be translated into a policy in a straightforward manner: process class 0 jobs while there are such jobs in the system (i.e., give priority to class 0) (Atkins and Chen, 1995). In fact, examining the remaining states of the system and

solving the corresponding linear programs leads to the (known to be optimal) $c\mu$ rule. Other ways of using the fluid model to determine a scheduling policy are discussed in Section 4.3.2.

In many cases, it can be shown that the fluid model is not just an approximation of the system of interest, but rather a formal limit of a sequence of scaled systems as the initial number of jobs in the system goes to infinity. Formal fluid models are of particular importance in stability analysis of queueing systems (Dai, 1995; Dai and Meyn, 1995). Stability analysis consists of identifying conditions under which the number of jobs in the system is guaranteed to remain bounded over time.[21] In the queueing literature, understanding of the stability of a system is considered to be a precursor to more detailed performance questions (Kumar and Meyn, 1995). Moreover, it has been shown that a system may be stable under one scheduling discipline, but not another (Kumar and Meyn, 1995; Bramson, 1994). In spite of its importance as a measure of long-run performance, stability of periodic scheduling approaches has only recently been introduced into the dynamic scheduling literature (Terekhov et al., 2012b). In particular, Terekhov et al. (2012b) and Terekhov (2013) show the stability of a method based on periodic makespan optimization in two flow shop systems, while Tran et al. (2013) and Terekhov et al. (2014) show stability of periodic scheduling methods in dynamic parallel machine settings.

While several fluid-model heuristics have been developed in the literature, their performance has not been empirically compared to predictive-reactive scheduling methods. For the parallel machine scheduling problem, Tran (2011) and Tran et al. (2013) compare a round-robin policy, which is derived from the solution of a fluid-model linear program, with a scheduling approach based on periodic makespan minimization. Studies of this type for other scheduling problems are necessary in order to understand the strengths and weaknesses of queueing and scheduling approaches, and to create effective hybrids, as is discussed by Terekhov (2013).

4.3.1.2 The Achievable Region Method The achievable region method is a mathematical programming approach for solving stochastic control optimization problems (Federgruen and Groenevelt, 1988; Bertsimas and Niño-Mora, 1996; Stidham, 2002). Similar to the approximations discussed above, the achievable region method aims to create a simpler representation of the scheduling problem of interest; instead of scaling time and the state space, it maps the problem to a lower-dimensional space of performance measures. In traditional mathematical programming formulations of scheduling problems, the goal is typically to determine the start times of the jobs or their positions in the processing sequence on a machine, with constraints being expressed in terms of these vari-

[21] In contrast, in the predictive-reactive literature, a predictive schedule is called *stable* if it does not change much as uncertainty is realized (Bidot et al., 2009). Similarly, in scheduling under uncertainty, stability analysis concerns the identification of the range of values that the processing times may take while the given schedule remains optimal (Sotskov et al., 2010).

ables. In the achievable region method, on the contrary, the decision variables are the performance measures for each class, and constraints are written in terms of these measures. Therefore, while the traditional methodology in scheduling is to find the schedule that optimizes the performance of the system, the achievable region approach is based on finding the optimal performance measure first, and then determining the corresponding scheduling policy. The idea of posting constraints on the performance that is achievable by a policy is similar to that of adding bounds on the quality of the schedule used in the scheduling literature.

More formally, consider as before a queueing system with $|\mathcal{K}|$ classes. Let u be a *control rule* that determines how each machine's time is allocated among the arriving job requests. Denote the set of all *admissible* control rules by $\mathcal{U}$. Although the precise definition of admissibility depends on the specific problem being addressed (Stidham, 2002), it is generally required that all control policies in $\mathcal{U}$ are *non-anticipative* (so that a decision can be based only on the current state of the problem and its history) and *non-idling* (so that a machine is not allowed to be idle if there are jobs waiting to be processed on this machine) (Dacre et al., 1999).

Define a $|\mathcal{K}|$-dimensional *system performance vector* $\boldsymbol{v}^u = (v_0^u, v_1^u, \ldots, v_{|\mathcal{K}|-1}^u)$ associated with every control policy u, where each v_k^u is the expected value of the performance measure for class k (Dacre et al., 1999). For example, if the goal of the problem is to minimize a weighted combination of the expected waiting times for each class, then the performance vector will consist of $|\mathcal{K}|$ expected waiting times, one for each class (Federgruen and Groenevelt, 1988). The set of all admissible performance vectors, $\Upsilon = \{\boldsymbol{v}^u, u \in \mathcal{U}\}$, is called the *performance space* or the *achievable region* of the problem. Frequently, the achievable region can be (at least partially) characterized by constraints derived from conservation laws, which state that the amount of work in the system due to a job class k under any policy is at least as much as the amount of class k work under the policy that gives this class priority over all other classes processed by a particular machine.

Suppose that the cost of running the system under the control u is denoted $c(\boldsymbol{v}^u)$. The scheduling problem of interest is therefore to find a rule, u^{OPT}, that would state how job classes should be assigned to machines in order to optimize the cost of running the system. More formally, this problem is stated by Dacre et al. (1999) as

$$Z^{OPT} = \inf_{u \in \mathcal{U}} \{c(\boldsymbol{v}^u)\}. \tag{36}$$

Alternatively, given Υ, u^{OPT} can be determined by solving the problem

$$Z^{OPT} = \inf_{\boldsymbol{v} \in \Upsilon} \{c(\boldsymbol{v})\}. \tag{37}$$

In other words, instead of finding the control rule that achieves the smallest cost by solving problem (36), we can

first find the best possible performance vector by solving (37) and then determine the optimal control associated with this performance vector. Thus, the *achievable region approach* is composed of three steps (Dacre et al., 1999):

1. Identification of the performance space Υ,

2. Solution of the mathematical programming problem (37),

3. Derivation of the optimal control rule u from the solution of problem (37).

To illustrate this approach, we consider a simplification of our example problem (Figure 1) which occurs if we ignore station 2. Specifically, suppose we want to find a non-anticipative and non-idling scheduling policy u for minimizing the long-run holding costs in a two-class $M/M/1$ system. Jobs of class k arrive to the system according to a Poisson process with rate λ_k and are processed according to an exponential distribution with rate μ_k. For stability, the rate at which work arrives to the system, $\rho_0 + \rho_1 = \lambda_0/\mu_0 + \lambda_1/\mu_1$, is assumed to be strictly less than 1. The problem can be stated as

$$Z^{OPT} = \inf_{u \in \mathcal{U}} \{c_0 E_u(q_0) + c_1 E_u(q_1)\}, \tag{38}$$

where $E_u(q_k)$ is the expected steady-state number of class k jobs present in the system operating under policy u, and c_k is the holding cost per job of class k.

In steady state, the amount of work present in this system is not dependent on the control policy u. Thus, the following constraints may be derived for $u \in \mathcal{U}$:

$$\frac{E_u(q_0)}{\mu_0} + \frac{E_u(q_1)}{\mu_1} = \frac{\rho_0 \mu_0^{-1} + \rho_1 \mu_1^{-1}}{1 - \rho_0 - \rho_1}, \tag{39}$$

$$\frac{E_u(q_0)}{\mu_0} \geq \frac{\rho_0 \mu_0^{-1}}{1 - \rho_0}, \tag{40}$$

$$\frac{E_u(q_1)}{\mu_1} \geq \frac{\rho_1 \mu_1^{-1}}{1 - \rho_1}. \tag{41}$$

The first equation is an expression for the expected amount of work in the system in steady state. The second and third equations follow from the observation that the steady state amount of class 0 work is minimized by giving priority to class 0 over class 1, and vice versa. Defining $\upsilon_k^u = E_u(q_k)/\mu_k$, the problem can be written as

$$Z^{OPT} = \inf_{\boldsymbol{\upsilon} \in \Upsilon} \{c_0 \mu_0 \upsilon_0 + c_1 \mu_1 \upsilon_1\}, \tag{42}$$

where

$$\Upsilon \quad = \quad \{(\upsilon_0, \upsilon_1); \upsilon_0 \geq \frac{\rho_0 \mu_0^{-1}}{1-\rho_0}, \upsilon_1 \geq \frac{\rho_1 \mu_1^{-1}}{1-\rho_1}, \upsilon_0 + \upsilon_1 = \frac{\rho_0 \mu_0^{-1} + \rho_1 \mu_1^{-1}}{1-\rho_0-\rho_1}\}. \tag{43}$$

The minimum of this problem is attained at the point where class 0 has absolute priority over class 1 if $c_0\mu_0 \geq c_1\mu_1$ and at the point where class 1 has priority over class 0 if $c_0\mu_0 < c_1\mu_1$. Since it can be easily shown that the performance space $\Upsilon = \{(\upsilon_0^u, \upsilon_1^u), u \in \mathcal{U}\}$ is equal to the line segment given by Υ, it follows that the solution to the original control problem is the well-known $c\mu$-rule: in this case, process the job class with the largest $c_k\mu_k$ value first (Dacre et al., 1999). The reader is referred to Section 3 of the paper by Bertsimas et al. (1994) for an achievable region analysis of our example problem with two stations.

The idea of characterizing the region of achievable performance and writing the problem in terms of constraints on performance vectors does not appear to have been used in scheduling, and should be explored in the future. Alternatively, the constraints developed for the achievable region approach in queueing theory may be profitably integrated into dynamic scheduling models.

4.3.2 Translation Techniques

In order for the approximation/abstraction methods to be useful in practice, their solutions need to be translated into implementable policies. One approach, demonstrated above, is to derive policies that mimic the intuition provided by the solutions to Brownian, fluid, or achievable region models. However, there are two main issues with this approach: it is problem-specific, and there are multiple ways of implementing the same intuition, some of which may in reality perform better than the others (see the paper by Maglaras (2000) for a discussion). Such issues have motivated the study of general translation mechanisms and their performance guarantees. To date, the study of such mechanisms has mostly been linked to Brownian and fluid models, and the application of similar principles to derive translation techniques from the achievable region method requires additional investigation.

A general approach for translation, presented in the paper by Chen and Meyn (1999), is to initialize the MDP value iteration algorithm (see Section 4.1) from the solution of the fluid model. Another approach is to define an affine shift policy based on the fluid policy (Meyn, 1997). In particular, suppose that the policy obtained from the fluid model is "process queue j at machine m if $j = \underset{k:s(k)=m}{\arg\max}\{\boldsymbol{\phi}_k^T \mathbf{q}\}$", where $\arg\max$ defines the argument of the maximum, $\mathbf{q}$ is the vector of current queue lengths and $\boldsymbol{\phi}_k$ is a vector of ratio-thresholds in $\mathbb{R}_+^{|\mathcal{K}|}$ which are obtained from the fluid model. To implement the policy in the original network, Meyn (1997) proposes to use the policy "process queue j at machine σ if $j = \underset{k:s(k)=m}{\arg\max}\{\boldsymbol{\phi}_k^T (\mathbf{q} - \mathbf{q}^0)\}$", where $\mathbf{q}^0$ defines an affine shift of the linear

switching curve of the fluid policy, and is used to prevent starvation of resources; $\mathbf{q}^0$ can be set, for example, to the expected value of $\mathbf{q}$.

The rest of the general translation mechanisms can be classified as either discrete-review or continuous-review. In discrete-review methods, the time between two review points is typically large, while in continuous-review ones, it is made as small as possible. Discrete-review methods have the advantage of requiring fewer optimizations, while continuous-review methods should perform better because they take the most up-to-date information into account (Teh, 2000). Both types of methods are similar in nature to periodic scheduling approaches from the scheduling literature: they periodically observe the status of the system and solve a resource allocation problem that prescribes how much time should be devoted to each job class until the next review point. Unlike periodic methods from the scheduling literature, these approaches concern the allocation of capacity among job classes rather than individual jobs. Another difference from scheduling methods is that the resource allocation problem solved at each review point is usually modelled as a linear program (LP). Related ideas, referred to as sequential open-loop strategies (Nash and Weber, 1982) and model-predictive control, have been successfully used within the electrical engineering control literature (van Leeuwaarden et al., 2010; Skaf and Boyd, 2010). We do not discuss the connections between queueing theory and that literature in this paper. References on early work on translation approaches can be found in the paper by Meyn (2001).

It is important to note that when a queue is empty at the approximation level, its counterpart in the underlying queueing network is *not* necessarily empty. This issue motivates the extensive use of the notion of safety stocks within both discrete-review and continuous-review paradigms. Safety stocks state the amount of material that should be present in the queues in order to avoid starvation of resources and thus help maintain stability (Meyn, 2008). Since, as discussed in Section 4.3.1.1, stability has not received much attention in the scheduling literature, neither has the use of safety stocks. Interestingly though, in the scheduling literature, Branke and Mattfeld (2005) demonstrate that avoiding early idleness is important for achieving good long-run performance.

4.3.2.1 Discrete Review We discuss several discrete-review approaches: BIGSTEP, reward-based policies and trajectory-tracking policies. According to Meyn (2008, p.129), discrete-review methods are "the most natural technique to translate a policy based on the fluid model or some other idealized model for application in a physical network".

BIGSTEP The first discrete-review approach developed in the queueing literature is called BIGSTEP. It was proposed by Harrison (1996) as a translation mechanism from solutions of Brownian approximation models, discussed in Section 4.3.1.1, to tactical allocations of machine time. The BIGSTEP approach is defined by two parameters: l, the length of time between two review time points, and $\boldsymbol{\zeta} = (\zeta_0, \ldots, \zeta_{|\mathcal{K}|-1})^T$, the threshold (planned safety stock) parameter vector (Teh, 2000). At every review time point $\tau = 0, l, 2l, \ldots$, the current queue lengths $\mathbf{q}(\tau) = (q_0(\tau), \ldots, q_{|\mathcal{K}|-1}(\tau))^T$ are observed, and an LP is formulated. This LP is called the BIGSTEP planning problem. Its solution specifies the amount of time that should be spent on processing each job class in the period $[\tau, \tau + l]$, subject to the constraint that the number of jobs in each queue k cannot fall below ζ_k.

If the corresponding Brownian approximation model allows for a pathwise solution,[22] then the BIGSTEP planning problem at time τ is:

$$\begin{aligned}
\text{minimize} \quad & \mathbf{cq}(\tau + l) && (44)\\
\text{subject to } \mathbf{q}(\tau + l) &= \mathbf{q}(\tau) + \boldsymbol{\lambda} l - \mathbf{A}\mathbf{u}, && (45)\\
\mathbf{i}(\tau + l) &= l\mathbf{1} - \mathbf{C}\mathbf{u}, && (46)\\
\mathbf{q}(\tau + l) &\geq \boldsymbol{\zeta}, && (47)\\
\mathbf{i}(\tau + l) &\geq \mathbf{0}, \quad \mathbf{u} \geq 0, && (48)
\end{aligned}$$

where $\mathbf{u} = (u_0, \ldots, u_{|\mathcal{K}|-1})^T$ with u_k specifying the amount of time that should be spent on class k within the time period $[\tau, \tau + l]$, $\mathbf{i}(l) = (i_1(l), \ldots, i_{|\mathcal{M}|}(l))^T$ with $i_m(l)$ being the cumulative amount of idleness from time τ to $\tau + l$ for machine m, $\mathbf{1}$ is a column vector of all 1s, $\mathbf{0}$ is a column vector of all 0s, and the rest of the notation is defined as before. Since this model requires first-moment data only, and since most dynamic control problems allow for a pathwise solution, the BIGSTEP approach can be applied to a wide range of problems. If the Brownian approximation model does not have a pathwise solution, then a second term, representing the future expected cost of machine idleness, is added to the objective in Equation (44). This additional term is $\boldsymbol{\kappa}\mathbf{i}(l)$, where $\boldsymbol{\kappa}$ is the penalty rate vector obtained by solving the appropriate Brownian control problem (Teh, 2000). The reader should note the similarity between the BIGSTEP planning problem and the fluid model in Equations (27)–(31) when $\tau = 0$ and $\zeta = 0$. For an example of the application of the BIGSTEP approach to a parallel machine system, we refer the reader to the paper by Harrison (1998).

[22] A pathwise solution is a dynamic scheduling policy that, in the heavy traffic limit, minimizes the instantaneous cost rate at every time point with probability one (Teh, 2000).

The idea behind the BIGSTEP approach is to ensure that the number of jobs present in each of the buffers is large enough relative to the length of the planning horizon so that the actual sequence in which the jobs are processed within the next l time units is not important from a tactical viewpoint (Harrison, 1996). In order to implement these tactical allocations at the operational level, one can choose an arbitrary sequence in which to consider classes, and process each class for the amount of time specified by the LP solution; a simple policy such as *FCFS* can be used to sequence jobs within each class (Harrison, 1996). The assumption that the details of the sequencing of individual jobs are "irrelevant" may seem unintuitive from a scheduling perspective. However, one needs to keep in mind that the goal of BIGSTEP is to optimize the long-run performance of the system and that the typical queueing-theoretic assumption is that job processing times do not become known upon the jobs' arrival to the system. Naturally, in realistic applications, both short-run and long-run performance measures are of interest and at least some processing time information may be available, which motivates the study of combining the BIGSTEP approach with detailed scheduling models of each time period.

Reward-based Policies Maglaras (1999) extends the BIGSTEP approach to a family of discrete-review methods derived from dynamic reward functions, which associate a positive reward rate $r_k(\mathbf{q})$ for spending time on class k, given an appropriately normalized queue length vector $\mathbf{q}$. Unlike in the BIGSTEP method, the length of the review time period in the reward-based discrete-review methods grows as a function of the queue lengths.

In addition to the system characteristics (e.g., arrival rates), the reward-based discrete-review approach of Maglaras (1999) requires three inputs: a reward function $r(\cdot)$; a function $g(\cdot)$ for computing the nominal length of the planning period; and a $|\mathcal{K}|$-dimensional vector $\boldsymbol{\xi}$ that satisfies $\boldsymbol{\xi} > \boldsymbol{\mu}$ (component-wise), used to define safety stock levels. At each review point τ_j, the controller calculates the length of the review period and the target safety stock level to be achieved at the end of the period. A linear program which is equivalent to the BIGSTEP linear program stated above, with the exception of the objective function, is then solved to determine the amount of time that should be allocated to each class k until the next review time point. The complete algorithm from the paper by Maglaras (1999) for the case when the planning linear program is feasible is presented in Figure 2.

Trajectory-tracking Policies The trajectory-tracking family of policies is also an extension and a generalization of the BIGSTEP method (Maglaras, 2000), and also requires a function $g(\cdot)$ and a $|\mathcal{K}|$-dimensional vector $\boldsymbol{\xi}$. Additionally, it is defined by the trajectory mapping Ψ, which describes the desired behaviour to be tracked. One option for the trajectory map is the solution of the corresponding fluid model (Maglaras, 2000).

given $r(\cdot)$, $g(\cdot)$, $\boldsymbol{\xi}$
$j = 0$; $\tau_0 = 0$;
initial review period length $l_0 = ag(|\mathbf{q}(0)|)$,
where a is a small ($<< 1$) positive constant independent of $|\mathbf{q}(0)|$
repeat $\{$
1. Compute review period length, l, and safety stock levels, $\boldsymbol{\zeta}$:
   $\mathbf{q} := \mathbf{q}(\tau_j)$, $\tilde{\mathbf{q}} := \mathbf{q}/|\mathbf{q}(0)|$, $l := l_0 \vee g(|\mathbf{q}|)$, $\mathbf{r} := r(\tilde{\mathbf{q}})$, $\boldsymbol{\zeta} := \boldsymbol{\xi} l$.
2. Compute the nominal activity allocations $\mathbf{u}$ by solving the LP:
   maximize $\mathbf{r}^T\mathbf{u}$
   subject to $\mathbf{q} + \boldsymbol{\lambda} l - \mathbf{A}\mathbf{u} \geq \boldsymbol{\zeta}$, $\mathbf{u} \geq \mathbf{0}$, $\mathbf{C}\mathbf{u} \leq l\mathbf{1}$.
3. Form processing plan $\boldsymbol{\alpha}$ with idleness budget $\mathbf{i}$:
   $\alpha_k := \min\{\lfloor \mu_k u_k \rfloor, q_k\}$ for $k = 0, 1, \ldots, |\mathcal{K}| - 1$; $\mathbf{i} := l\mathbf{1} - \mathbf{C}\mathbf{u}$.
4. Execute $(\boldsymbol{\alpha}, \mathbf{i})$, which takes a length of time denoted by T^{exe}.
5. Update: $\tau_{j+1} := \tau_j + T^{exe}$; $j := j + 1$.

$\}$

Figure 2: Reward-based Discrete-Review Algorithm (adapted from the paper by Maglaras (1999)).

The trajectory-tracking approach to scheduling in a queueing system resembles the algorithm in Figure 2, with three major changes: at step 1, a target state, $\mathbf{z} = \boldsymbol{\zeta} + |\mathbf{q}(0)|\Psi(\frac{l}{|\mathbf{q}(0)|}; \max\{0, (\mathbf{q} - \boldsymbol{\zeta})\}/|\mathbf{q}(0)|)$, is chosen for the end of the review period; step 2 is replaced by calculations to find $\mathbf{u}$ that would allow the system to get to state $\mathbf{z}$; at step 3, the processing plan is defined as follows:

$$\alpha_k = \left\lfloor \mu_k u_k \right\rfloor, \quad k = 0, \ldots, |\mathcal{K}| - 1, \tag{49}$$

$$i_m = \max\{0, (l + \tau_{\boldsymbol{\zeta}} - (\mathbf{C}\mathbf{u})_k)\}, \quad m = 1, \ldots, |\mathcal{M}|, \tag{50}$$

where $\tau_{\boldsymbol{\zeta}}$ is the length of time needed to get from state $\mathbf{q} \wedge \boldsymbol{\zeta}$ to $\boldsymbol{\zeta}$ ($\wedge$ denotes the component-wise minimum). The resulting policies are asymptotically optimal under fluid scaling, and guaranteed to be stable if the traffic intensity of each station is less than one (Maglaras, 2000). A similar idea of trajectory-tracking policies is explored by Meyn (2001), resulting in a class of *feedback regulation* policies.

The idea of trajectory tracking has not been investigated in the scheduling literature. We believe it should be examined from two perspectives: firstly, predictive-reactive methods could be developed which aim to track the solution of the fluid model as do the trajectory-tracking policies in queueing; secondly, for static problems, there may be problem relaxations, such as preemptive versions of non-preemptive problems, that can be tracked.

4.3.2.2 Continuous Review We discuss several continuous-review approaches: discrete-review derived, trajectory-tracking, and maximum pressure policies.

Discrete-review Derived Policies The work of Teh (2000) extends the idea of the BIGSTEP method to discrete-review derived (DRD) continuous-review policies. As in the BIGSTEP method, the goal of a DRD policy is to determine the amount of time that should be devoted to processing a particular class of jobs in the coming time period of length l. However, in the BIGSTEP approach, the value of l needs to be large, while Teh (2000) proposes to make l as small as possible without forcing the time allocation variables to take on negative values. Moreover, the value of l may change when the allocation specified by the LP solved at each review point changes.

Trajectory-tracking Policies Bäuerle (2000) defines a class of *tracking* policies, which are constructed from the optimal fluid control rule in such a way that the scaled state process converges to the optimal fluid trajectory. Maglaras (2003), motivated by the trajectory-tracking idea from the area of model predictive control, proposes policies that track a target state $\mathbf{z}$ obtained by solving a Brownian control problem.

Paschalidis et al. (2003; 2004) define target-pursuing policies, which are similar to trajectory-tracking policies, but utilize the solution of an achievable region problem to set one of its parameters. At each time t and for a finite review interval Δt, a target-pursuing policy minimizes $\mathbf{E}[||\mathbf{q}(t+\Delta t) - \mathbf{z}||\ |\mathbf{q}(t)]$ for some norm $||\cdot||$ and target $\mathbf{z}$; that is, given the number of jobs in each class at time t, $\mathbf{q}(t)$, the policy minimizes the expectation (with respect to the probability distribution of $\mathbf{q}(t+\Delta t)$) of the norm of the difference between the number of jobs present in each class at the next review point and the target (Paschalidis et al., 2004). Paschalidis et al. (2004) show that setting $\mathbf{z} = \boldsymbol{\upsilon}^*$, where $\boldsymbol{\upsilon}^*$ is the lower bound on the optimal performance obtained via an achievable region approach, often results in good performance. For example, in the context of the problem of Figure 1, target-pursuing policies with optimized parameters are numerically shown to be within 2.3% of the best-performing policies for various loads for the weighted sum of mean queue lengths objective.

Maximum Pressure Policies Maximum pressure policies (Dai and Lin, 2005) are generalizations of the well-studied MaxWeight policy (Tassiulas and Ephremides, 1992; Tassiulas and Bhattacharya, 2000; Andrews et al., 2004; Stolyar, 2004). A maximum pressure policy looks at the system state at job completion and job arrival epochs, and determines the allocation vector $\mathbf{u} = (u_j)$ that maximizes the total network pressure (Dai and Lin, 2005). Each entry u_j specifies the proportion of time that should be spent on processing job j (by the appropriate machine(s)) that remains valid until the next review point. More formally, one has to choose $\mathbf{u} \in \mathcal{E}(t)$ which maximizes $\nu(\mathbf{u}, \mathbf{q}(t)) = \mathbf{q}(t) \cdot \mathbf{R}\mathbf{u}$, where $\mathbf{q}(t)$ is the vector of buffer levels at time t, $\mathbf{R} = (R_{kj})$ is the input-output

matrix in which each entry R_{kj} is interpreted as the average amount of buffer k material consumed per unit of job j, $\mathcal{E}(t)$ is the set of extreme points of the feasible region of all possible maximum pressure policies, and $\cdot$ denotes the inner product between the two vectors. The set $\mathcal{E}(t)$ depends on whether resources can process one or several jobs at a time (i.e., whether processor splitting is allowed or not) and on whether preemptions are allowed or not. Maximum pressure policies are different from the above-mentioned policies since they directly specify which job should receive processing next. They are semi-local since the sequencing decisions for each machine are based on both the state of the buffers for which the machine is responsible and the state of the downstream buffers. One of the advantages is that the policy does not use any arrival rate information, which can be hard to estimate in practice (Dai and Lin, 2005).

The work discussed in this section on alternative representations is aimed at deriving policies which are asymptotically optimal and stable, and, given the long-run nature of these objectives, they prescribe the proportion of each server's capacity that should be spent on processing a particular job class. In real scheduling problems, one typically needs to address short-run performance measures in addition to long-run ones. Therefore, it is worthwhile to investigate different operational-level scheduling policies that can also respect the tactical allocations provided by methods discussed in this section. For example, the solutions to approximations/abstractions may be used in the form of bounds when operational-level schedules need to be constructed. Moreover, a model that specifies high-level allocations could be used as part of a problem decomposition method such as the logic-based Benders method of Hooker (2005); the work of Aramon Bajestani and Beck (2013) and Aramon Bajestani (2013, Chapter 6) on the use of Benders decomposition in problems with maintenance and scheduling decisions would be useful in the study of this direction.

5 Future Work on the Integration of Queueing Theory and Scheduling

In this section, we summarize future work on the integration of queueing theory and scheduling that was mentioned above as well as proposing some additional future work directions. We classify these ideas into three levels at which we believe integration can take place: conceptual, theoretical and algorithmic.

5.1 Conceptual Level

The conceptual level of integration of queueing and scheduling focuses on combining problem settings, concepts and assumptions from the two areas. Specifically, the following ideas should be investigated in the future:

- continuation of the work by Harchol-Balter (2011), Nuyens et al. (2008) and Wierman et al. (2005) on using queueing analysis to derive theoretical performance guarantees for dispatching rules under particular distributional assumptions (Section 4.2.1);

- development of new priority queueing models based on dispatching rules (e.g., composite dispatching rules such as the Apparent Tardiness Cost heuristic (Pinedo, 2009)) and their analysis (Section 4.2.1);

- development of a general framework for integrating queueing and scheduling based on the observation that *any* scheduling algorithm can be represented as a function of M, P and A, where M is the decision mode, P is a priority function, and A is an arbitration rule (Jaiswal, 1982; Ruschitzka and Fabry, 1977) (Section 4.2.1);

- investigation of the applicability of results from the study of fair sequences (Kubiak, 2004) to the study of optimization of the polling order (Section 4.2.2);

- study of problems that require optimization of polling order together with optimization of within-queue sequencing (e.g., simultaneously minimizing the expected length of a production cycle and the total job tardiness in a manufacturing facility) (Section 4.2.2);

- improvement of polling system performance by employing scheduling approaches to optimize within-queue scheduling (Section 4.2.2);

- investigation of how polling models can represent systems with different objectives at different decision-making levels (Section 4.2.2);

- further examination of the link between lot-sizing and polling systems (Winands, 2007), and the use of lot-sizing or batch scheduling methods for optimization of polling order within polling systems (Section 4.2.2);

- investigation of the use of polling models with precedence constraints (Khamisy et al., 1992) for modelling scheduling problems with precedences and the task management problem of Myers et al. (2007) (Section 4.2.2);

- study of the relationship between the restless bandit problem and stochastic variations of resource-constrained project scheduling discussed by Mercier and Van Hentenryck (2008) (Section 4.2.4).

In general, queueing theory and scheduling make differing assumptions regarding the information that is available for making decisions. In queueing theory, it is typically assumed that processing time distributions are known, but the actual processing time of a job is not known until it is finished. In classical scheduling, on the contrary, the common assumption is that a job's processing time becomes known with certainty upon its arrival to the system. Thus, following the work by Tran (2011), a general future work direction is to examine a wide range of problems under a combination of queueing and scheduling assumptions, that is, assuming both knowledge of distributional information and knowledge of individual job characteristics. From the theoretical perspective, there has not been much previous work on this "middle ground" between queueing and scheduling assumptions, although there has been a significant amount of work on the extremes (i.e., knowledge of distributions only, or knowledge of exact processing times only). From the practical perspective, there exist applications (e.g., in manufacturing and computer processing) in which distributional information can be derived from historical data and job processing times can be well estimated upon their arrival to the system.

5.2 Theoretical Level

The theoretical level of integration focuses on combining theoretical notions from queueing and scheduling. The following ideas for future work were proposed above:

- investigation of the steady-state performance of composite dispatching rules (Section 4.2.1);
- application of descriptive results from polling systems to the development of bounds on optimal schedules and within scheduling algorithms (Section 4.2.2).

5.3 Algorithmic Level

As discussed in Section 4, scheduling algorithms developed within queueing theory are typically concerned with the allocation of resources to job classes rather than with detailed sequencing decisions. They are likely to optimize long-run objectives but may perform poorly for a given short time horizon. Scheduling methods, in contrast, focus on sequencing and optimization of short-run performance but may be myopic. Thus, combining algorithmic components from queueing theory and scheduling could lead to the development of hybrid algorithms for effectively addressing dynamic scheduling problems. The following ideas for integration of queueing and scheduling on the algorithmic level become evident from our review:

- empirical comparison of the performance of queueing policies (e.g., priority policies, fluid model-based heuristics), predictive-reactive scheduling methods, online stochastic combinatorial optimization methods and hybrid algorithms, similar in style to the algorithmic work of Tran (2011) (Sections 4.2.1, 4.3.1.1, 4.3.2);

- examination of the MDP model as a meta-framework for the construction of queueing/scheduling hybrids: if a queueing algorithm and a scheduling algorithm are chosen, and the state of the problem is compactly represented, then we can define an action as the choice to follow the queueing algorithm or the choice to follow the scheduling algorithm until the next decision time point (Section 4.1);

- investigation of whether it is useful to model part or all of a dynamic scheduling problem as a bandit problem and to incorporate Gittins indices into the constraints or the objective function of the models used to construct predictive schedules (Section 4.2.4);

- investigation of the applicability of the achievable region approach in scheduling (e.g., for derivation of performance bounds) (Section 4.3.1.2);

- derivation of new constraints for the achievable region approach based on techniques from scheduling (Section 4.3.1.2);

- investigation of operational-level scheduling policies that can also respect the tactical allocations provided by queueing methods (e.g., Markov Decision Process approaches, fluid models, Brownian models) (Sections 4.1, 4.3.1.1, 4.3.2), similar to the investigation of Aramon Bajestani et al. (2014) which combines higher-level maintenance decisions with lower-level operational ones;

- investigation of the idea of tracking for scheduling: firstly, predictive-reactive methods could be developed which aim to track the solution of the fluid model as do the trajectory-tracking policies in queueing, and, secondly, for static problems, there may be problem relaxations, such as preemptive versions of non-preemptive problems, that can be tracked (Section 4.3.2);

- use of a model that specifies high-level resource allocations as part of a problem decomposition method such as the logic-based Benders method of Hooker (2005) (Section 4.3.2); the work of Aramon Bajestani and Beck (2011, 2013) on the use of Benders decomposition in problems with maintenance and scheduling decisions would be useful in the study of this direction.

6 Conclusion

We surveyed the queueing theory literature related to scheduling, dividing it into three categories. The first is based on MDP models, which are, theoretically, powerful enough to represent any scheduling problem. However, even when focusing on classes rather than individual jobs, MDP models become intractable as the size of the problem (i.e., the number of classes) increases. To overcome the difficulties of large MDP models, the queueing literature has chosen either to restrict the types of policies that are considered or to use alternative problem representations based on approximations or abstractions.

Section 4.2 reviewed the second category of models: those possessing a special structure. These include priority queues, polling systems, vacation models and bandit models. Section 4.3 presented the general framework for the use of alternative problem representations which consists of four steps: modelling and development of the approximation/abstraction, solution of this approximation/abstraction, derivation of an implementable scheduling policy based on this solution, and analysis of the optimality of the resulting policy. The approximations/abstractions used in the first two steps include Brownian models, fluid models and the achievable region approach. Methods for the derivation of implementable scheduling policies are presented in Section 4.3.2.

Throughout this review, we have indicated various directions for future work on the integration of queueing and scheduling. For instance, one common theme throughout this paper is that, since queueing-theoretic approaches cannot deal with individual job characteristics, scheduling can be used to sequence jobs within each job class. Doing so is especially important in applications where historical data is available and actual job durations can be accurately estimated upon their arrival to the system. An extensive list of ideas for integrating queueing and scheduling was presented in Section 5.

Real scheduling problems are dynamic, stochastic and combinatorial in nature. While classical scheduling has focused on the combinatorial aspects and short-term objectives, queueing theory has been concerned with stochastic properties and performance of the system over the long term. Thus, from the dynamic scheduling perspective, the strengths of queueing theory and scheduling can be seen as being complementary. We believe that integrating the two may allow us to develop both a better understanding of, and more effective solution techniques for, scheduling in dynamic and stochastic environments, because we can more effectively reason about the long run using queueing theory and about the short run using scheduling methods. We have published several papers that investigate and provide support for this belief (Terekhov et al., 2012b; Tran et al., 2013; Terekhov et al., 2014).

Acknowledgements

The authors would like to thank the reviewers for their comments, which helped improve the paper.

This research has been supported by the Natural Sciences and Engineering Research Council of Canada, the Canadian Foundation for Innovation, the Ontario Research Fund, the Ontario Ministry for Research and Innovation, the Ireland Industrial Development Agency, Alcatel-Lucent, Microway Inc., IBM ILOG, University of Toronto School of Graduate Studies Doctoral Completion Award, and the Department of Mechanical and Industrial Engineering at the University of Toronto.

References

Abbasi-Yadkori, Y., Bartlett, P., Malek, A., 2014. Linear programming for large-scale Markov decision problems, in: Proceedings of the 31st International Conference on Machine Learning, pp. 496–504.

Adan, I., Resing, J., 2002. Queueing Theory. Department of Mathematics and Computing Science, Eindhoven University of Technology. Available online at http://www.win.tue.nl/˜ iadan/queueing.pdf.

Ahn, H.S., Duenyas, I., Lewis, M.E., 2002. Optimal control of a two-stage tandem queuing system with flexible servers. Probability in the Engineering and Informational Sciences 16, 453–469.

Altman, E., Yechiali, U., 1993. Cyclic Bernoulli polling. Mathematical Methods of Operations Research 38, 55–76.

Andradóttir, S., Ayhan, H., Down, D.G., 2003. Dynamic server allocation for queueing networks with flexible servers. Operations Research 51, 952–968.

Andrews, M., Kumaran, K., Ramanan, K., Stolyar, A., Vijayakumar, R., Whiting, P., 2004. Scheduling in a queueing system with asynchronously varying service rates. Probability in the Engineering and Informational Sciences 18, 191–217.

Antunes, N., Fricker, C., Roberts, J., 2011. Stability of multi-server polling system with server limits. Queueing Systems 11, 229–235.

Aramon Bajestani, M., 2013. Integrating Maintenance Planning and Production Scheduling: Making Operational Decisions with a Strategic Perspective. Ph.D. thesis. Department of Mechanical and Industrial Engineering, University of Toronto.

Aramon Bajestani, M., Banjevic, D., Beck, J.C., 2014. Integrated maintenance planning and production scheduling with markovian deteriorating machine conditions. International Journal of Production Research To Appear.

Aramon Bajestani, M., Beck, J.C., 2011. Scheduling an aircraft repair shop, in: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS'11), pp. 10–17.

Aramon Bajestani, M., Beck, J.C., 2013. Scheduling a dynamic aircraft repair shop with limited repair resources. Journal of Artificial Intelligence Research 47, 35–70.

Atkins, D., Chen, H., 1995. Performance evaluation of scheduling control of queueing networks: Fluid model heuristics. Queueing Systems 21, 391–413.

Aytug, H., Lawley, M., McKay, K., Mohan, S., Uzsoy, R., 2005. Executing production schedules in the face of uncertainties: A review and future directions. European Journal of Operational Research 161, 86–110.

Baccelli, F., Liu, Z., 1990. On the execution of parallel programs on multiprocessor systems – a queuing theory approach. Journal of the Association for Computing Machinery 37, 373–414.

Baccelli, F., Makowski, A.M., 1989. Queueing models for systems with synchronization constraints. Proceedings of the IEEE 77, 138–161.

Baccelli, F., Massey, W.A., Towsley, D., 1989. Acyclic fork-join queuing networks. Journal of the Association for Computing Machinery 36, 615–642.

Baker, K.R., Trietsch, D., 2009. Principles of Sequencing and Scheduling. John Wiley & Sons.

Bandi, C., Bertsimas, D., 2012. Tractable stochastic analysis in high dimensions via robust optimization. Mathematical programming 134, 23–70.

Bandi, C., Bertsimas, D., Youssef, N., 2012. Robust queueing theory. Submitted for publication .

Basnet, C., Mize, J.H., 1994. Scheduling and control of flexible manufacturing systems: a critical review. International Journal of Computer Integrated Manufacturing 7, 340–355.

Bäuerle, N., 2000. Asymptotic optimality of tracking policies in stochastic networks. Annals of Applied Probability 10, 1065–1083.

Beck, J.C., Wilson, N., 2007. Proactive algorithms for job shop scheduling with probabilistic durations. Journal of Artificial Intelligence Research 28, 183–232.

Beekhuizen, P., Denteneer, D., Resing, J., 2008. Reduction of a polling network to a single node. Queueing Systems 58, 303–319.

Ben-Tal, A., El Ghaoui, L., Nemirovski, A., 2009. Robust optimization. Princeton University Press.

Bertsekas, D.P., 2005. Dynamic Programming and Optimal Control: Volume I. 3 ed., Athena Scientific.

Bertsekas, D.P., 2007. Dynamic Programming and Optimal Control: Volume II. 3 ed., Athena Scientific.

Bertsimas, D., 1995. The achievable region method in the optimal control of queueing systems: Formulations, bounds, policies. Queueing Systems 21, 337–389.

Bertsimas, D., Gamarnik, D., 1999. Asymptotically optimal algorithms for job shop scheduling and packet routing. Journal of Algorithms 33, 296–318.

Bertsimas, D., Gamarnik, D., Rikun, A.A., 2011. Performance analysis of queueing networks via robust optimization. Operations research 59, 455–466.

Bertsimas, D., Gamarnik, D., Sethuraman, J., 2003. From fluid relaxations to practical algorithms for high-multiplicity job-shop scheduling: the holding cost objective. Operations Research 51, 798–813.

Bertsimas, D., Niño-Mora, J., 1996. Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems. Mathematics of Operations Research 21, 257–306.

Bertsimas, D., Pachamanova, D., Sim, M., 2004. Robust linear optimization under general norms. Operations Research Letters 32, 510–516.

Bertsimas, D., Paschalidis, I.C., Tsitsiklis, J.N., 1994. Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance. The Annals of Applied Probability 4, 43–75.

Bertsimas, D., Paschalidis, I.C., Tsitsiklis, J.N., 1995. Branching bandits and Klimov's problem: Achievable region and side constraints. IEEE Transactions on Automatic Control 40, 2063–2075.

Bertsimas, D., Sethuraman, J., 2002. From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective. Mathematical Programming 92, 61–102.

Bertsimas, D., Sim, M., 2004. The price of robustness. Operations Research 52, 35–53.

Bertsimas, D.J., Xu, H., 1993. Optimization of polling systems and dynamic vehicle routing problems on networks. Technical Report OR 283-93. Massachusetts Institute of Technology, Operations Research Center.

Bidot, J., Vidal, T., Laborie, P., Beck, J.C., 2009. A theoretic and practical framework for scheduling in a stochastic environment. Journal of Scheduling 12, 315–344.

Blanc, J.P.C., van der Mei, R.D., 1995. Optimization of polling systems with Bernoulli schedules. Performance Evaluation 22, 139–158.

Boon, M.A.A., Adan, I.J.B.F., 2009. Mixed gated/exhaustive service in a polling model with priorities. Queueing Systems 63, 383–399.

Boon, M.A.A., Van der Mei, R.D., Winands, E.M.M., 2011. Applications of polling systems. Surveys in Operations Research and Management Science 16, 67–82.

Borst, S.C., 1995. Polling systems with multiple coupled servers. Queueing systems 20, 369–393.

Borst, S.C., Boxma, O.J., Harink, J.H.A., Huitema, G.B., 1994. Optimization of fixed time polling schemes. Telecommunication Systems 3, 31–59.

Bose, S.K., 2002. An Introduction to Queueing Systems. Springer.

Boudoukh, T., Penn, M., Weiss, G., 2001. Scheduling jobshops with some identical or similar jobs. Journal of Scheduling 4, 177–199.

Boxma, O., Van Der Wal, J., Yechiali, U., 2008. Polling with batch service. Stochastic Models 24, 604–625.

Boxma, O.J., Groenendijk, W.P., 1987. Pseudo-conservation laws in cyclic-service systems. Journal of Applied Probability 24, 949–964.

Boxma, O.J., Levy, H., Weststrate, J.A., 1989. Optimization of polling systems. Technical Report BS-R8932. Department of Operations Research, Statistics, and System Theory, CWI Centre for Mathematics and Computer Science, Amsterdam, The Netherlands.

Boxma, O.J., Levy, H., Weststrate, J.A., 1991. Efficient visit frequencies for polling tables: minimization of waiting cost. Queueing Systems 9, 133–162.

Boxma, O.J., Levy, H., Weststrate, J.A., 1993. Efficient visit orders for polling systems. Performance Evaluation 18, 103–123.

Boyd, S., Barratt, C., 1991. Linear controller design: limits of performance. Prentice Hall Englewood Cliffs, NJ.

Bramson, M., 1994. Instability of FIFO queueing networks. The Annals of Applied Probability , 414–431.

Branke, J., Mattfeld, D.C., 2002. Anticipatory scheduling for dynamic job shop problems, in: Proceedings of the ICAPS'02 Workshop on On-line Planning and Scheduling, pp. 3–10.

Branke, J., Mattfeld, D.C., 2005. Anticipation and flexibility in dynamic scheduling. International Journal of Production Research 43, 3103–3129.

Browne, S., Weiss, G., 1992. Dynamic priority rules when polling with multiple parallel servers. Operations Research Letters 12, 129–137.

Browne, S., Yechiali, U., 1989a. Dynamic priority rules for cyclic-type queues. Advances in Applied Probability 21, 432–450.

Browne, S., Yechiali, U., 1989b. Dynamic routing in polling systems. Teletraffic Science for New Cost-Effective Systems, Networks and Services, ITC-12 , 1455–1466.

Budhiraja, A., Ghosh, A.P., 2005. A large deviations approach to asymptotically optimal control of crisscross network in heavy traffic. The Annals of Applied Probability 15, 1887–1935.

Buzacott, J.A., Shanthikumar, J.A., 1993. Stochastic Models of Manufacturing Systems. Prentice Hall.

Buzacott, J.A., Shanthikumar, J.G., 1985. On approximate queueing models of dynamic job shops. Management Science , 870–887.

Buzacott, J.A., Yao, D.D., 1986. Flexible manufacturing systems: a review of analytical models. Management Science 32, 890–905.

Chan, W.K., Schruben, L., 2008. Optimization models of discrete-event system dynamics. Operations Research 56, 1218–1237.

Chao, X., Zhao, Y.Q., 1998. Analysis of multi-server queues with station and server vacations. European Journal of Operational Research 110, 392–406.

Chen, H., Mandelbaum, A., 1991. Discrete flow networks: bottleneck analysis and fluid approximations. Mathematics of Operations Research , 408–446.

Chen, H., Mandelbaum, A., 1994a. Hierarchical modeling of stochastic networks, part I: Fluid models, in: Yao, D.D. (Ed.), Stochastic modeling and analysis of manufacturing systems. Springer. Springer Series in Operations Research, pp. 47–105.

Chen, H., Mandelbaum, A., 1994b. Hierarchical modeling of stochastic networks, part II: Strong approximations, in: Yao, D.D. (Ed.), Stochastic Modeling and Analysis of Manufacturing Systems. Springer. Springer Series in Operations Research, pp. 107–131.

Chen, H., Yang, P., Yao, D.D., 1994. Control and scheduling in a two-station queueing network: Optimal policies and heuristics. Queueing Systems 18, 301–332.

Chen, H., Yao, D.D., 1993. Dynamic scheduling of a multiclass fluid network. Operations Research 41, 1104–1115.

Chen, H., Yao, D.D. (Eds.), 2001. Fundamentals of Queueing Networks. Springer.

Chen, R.R., Meyn, S., 1999. Value iteration and optimization of multiclass queueing networks. Queueing Systems 32, 65–97.

Chevalier, P.B., Wein, L.M., 1993. Scheduling networks of queues: heavy traffic analysis of a multistation closed network. Operations Research 41, 743–758.

Chrétienne, P., Coffman Jr., E.G., Lenstra, J.K., Liu, Z. (Eds.), 1995. Scheduling Theory and Its Applications. John Wiley & Sons Ltd.

Cicirello, V.A., Smith, S.F., 2005. The max K-armed bandit: A new model of exploration applied to search heuristic selection, in: Proceedings of Twentieth National Conference on Artificial Intelligence (AAAI'05), pp. 1355–1361.

Conway, R.W., Maxwell, W.L., Miller, L.W., 1967. Theory of Scheduling. Addison-Wesley.

Cox, D.R., Smith, W.L., 1961. Queues. Methuen.

Crabill, T., Gross, D., Magazine, M., 1977. A classified bibliography of research on optimal design and control of queues. Operations Research 25, 219–232.

Dacre, M., Glazebrook, K., Niño-Mora, J., 1999. The achievable region approach to the optimal control of stochastic systems. Journal of the Royal Statistical Society: Series B 61, 747–791.

Dai, J.G., 1995. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. The Annals of Applied Probability 5, 49–77.

Dai, J.G., Lin, W., 2005. Maximum pressure policies in stochastic processing networks. Operations Research 53, 197–218.

Dai, J.G., Meyn, S.P., 1995. Stability and convergence of moments for multiclass queueing networks via fluid limit models. IEEE Transactions on Automatic Control 40, 1889–1904.

Dai, J.G., Prabhakar, B., 2000. The throughput of data switches with and without speedup, in: Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00), IEEE. pp. 556–564.

Dai, J.G., Weiss, G., 2002. A fluid heuristic for minimizing makespan in job shops. Operations Research 50, 692–707.

Daniels, R., Carrillo, J., 1997. β-Robust scheduling for single-machine systems with uncertain processing times. IIE Transactions 29, 977–985.

Davenport, A.J., Beck, J.C., 2000. A Survey of Techniques for Scheduling with Uncertainty. Technical Report. University of Toronto. Available at: http://www.tidel.mie.utoronto.ca/publications.php.

Davenport, A.J., Gefflot, C., Beck, J.C., 2001. Slack-based techniques for robust schedules, in: Proceedings of the Sixth European Conference on Planning (ECP'01).

Day, M.V., 2006. Boundary-influenced robust controls: two network examples. ESAIM: Control, Optimisation and Calculus of Variations 12, 662–698.

Derman, C., 1970. Finite State Markovian Decision Processes. Academic Press.

Desai, V.V., Farias, V.F., Moallemi, C.C., 2012. Approximate dynamic programming via a smoothed linear program. Operations Research 60, 655–674.

Dieker, A.B., 2010. Reflected Brownian motion. Wiley Encyclopedia of Operations Research and Management Science .

Doshi, B., 1986. Queueing systems with vacations – a survey. Queueing Systems 1, 29–66.

Down, D., 1998. On the stability of polling models with multiple servers. Journal of Applied Probability 35, 925–935.

Drekic, S., Stanford, D.A., 2000. Threshold-based interventions to optimize performance in preemptive priority queues. Queueing Systems 35, 289–315.

Dubois, D., 1983. A mathematical model of a flexible manufacturing system with limited in-process inventory. European Journal of Operational Research 14, 66–78.

Dumitriu, I., Tetali, P., Winkler, P., 2003. On playing golf with two balls. SIAM Journal on Discrete Mathematics 16, 604–615.

Dupuis, P., 2003. Explicit solution to a robust queueing control problem. SIAM Journal on Control and Optimization 42, 1854–1875.

El-Taha, M., Stidham, S.J., 1999. Sample-Path Analysis of Queueing Systems. Kluwer.

Ephremides, A., Varaiya, P., Walrand, J., 1980. A simple dynamic routing problem. Automatic Control, IEEE Transactions on 25, 690–693.

Fan-Orzechowski, X., Feinberg, E.A., 2007. Optimality of randomized trunk reservation for a problem with multiple constraints. Probability in the Engineering and Informational Sciences 21, 189–200.

de Farias, D.P., Van Roy, B., 2003. The linear programming approach to approximate dynamic programming. Operations Research 51, 850–865.

de Farias, D.P., Weber, T., 2008. Choosing the cost vector of the linear programming approach to approximate dynamic programming, in: Proceedings of the 47th IEEE Conference on Decision and Control (CDC'08), IEEE. pp. 67–72.

Federgruen, A., Groenevelt, H., 1988. Characterization and optimization of achievable performance in general queueing systems. Operations Research 36, 733–741.

Fournier, L., Rosberg, Z., 1991. Expected waiting times in polling systems under priority disciplines. Queueing Systems 9, 419–440.

Fuhrmann, S.W., 1984. A note on the M/G/1 queue with server vacations. Operations Research 32, 1368–1373.

Gagliolo, M., Schmidhuber, J., 2007. Learning restart strategies, in: Proceedings of the of the Twenty First International Joint Conference on Artificial Intelligence (IJCAI'07), pp. 792–797.

Garcia, C.E., Prett, D.M., Morari, M., 1989. Model predictive control: theory and practicea survey. Automatica 25, 335–348.

Gaujal, B., Hordijk, A., van der Laan, D., 2007. On the optimal open-loop control policy for deterministic and exponential polling systems. Probability in the Engineering and Informational Sciences 21, 157–187.

Ghallab, M., Nau, D., Traverso, P., 2004. Automated Planning: Theory and Practice. Morgan Kaufman.

Gittins, J.C., 1979. Bandit processes and dynamic allocation indices. Journal of the Royal Statistical Society: Series B (Methodological) 41, 148–177.

Glasserman, P., Yao, D.D., 1994. Monotone optimal control of permutable GSMPs. Mathematics of Operations Research 19, 449–476.

Glazebrook, K.D., Owen, R.W., 1995. Gittins-index heuristics for research planning. Naval Research Logistics 42, 1041–1062.

Goldberg, H.M., 1977. Analysis of the earliest due date scheduling rule in queueing systems. Mathematics of Operations Research 2, 145–154.

Govil, M.K., Fu, M.C., 1999. Queueing theory in manufacturing: A survey. Journal of Manufacturing Systems 18, 214–240.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5, 287–326.

Grassmann, W., Chen, X., Kashyap, B.R.K., 2001. Optimal service rates for the state-dependent M/G/1 queues in steady-state. Operations Research Letters 29, 57–63.

Grassmann, W.K., 1977. Transient solutions in Markovian queueing systems. Computers & Operations Research 4, 47–53.

Gross, D., Shortle, J.F., Thompson, J.M., Harris, C.M., 2008. Fundamentals of queueing theory. John Wiley & Sons, Inc.

Gupta, A.K., Sivakumar, A.I., 2006. Job shop scheduling techniques in semiconductor manufacturing. The International Journal of Advanced Manufacturing Technology 27, 1163–1169.

Gurvich, I., Whitt, W., 2009. Scheduling flexible servers with convex delay costs in many-server service systems. Manufacturing & Service Operations Management 11, 237–253.

Hajek, B., 1984. Optimal control of two interacting service stations. IEEE Transactions on Automatic Control 29, 491–499.

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

Hall, L.A., 1997. Approximation algorithms for scheduling, in: Hochbaum, D.S. (Ed.), Approximation algorithms for NP-hard problems. PWS Publishing Company. chapter 1, pp. 1–45.

Harchol-Balter, M., 2011. Queueing disciplines. Wiley Encyclopedia of Operations Research and Management Science .

Hariharan, R., Moustafa, M.S., Stidham, S.J., 1996. Scheduling in a multi-class series of queues with deterministic service times. Queueing Systems 24, 83–99.

Harrison, J.M., 1975. Dynamic scheduling of a multiclass queue: Discount optimality. Operations Research 23, 270–282.

Harrison, J.M., 1988. Brownian models of queueing networks with heterogeneous customer populations, in: Fleming, W., Lions, P.L. (Eds.), Stochastic Differential Systems, Stochastic Control Theory and Applications, Springer-Verlag. pp. 147–186.

Harrison, J.M., 1996. The BIGSTEP approach to flow management in stochastic processing networks, in: Kelly, F.P., Zachary, S., Ziedins, I. (Eds.), Stochastic Networks: Theory and Applications. Oxford University Press. chapter 4, pp. 57–90.

Harrison, J.M., 1998. Heavy traffic analysis of a system with parallel servers: Asymptotic optimality of discrete-review policies. The Annals of Applied Probability 8, 822–848.

Harrison, J.M., 2003. A broader view of Brownian networks. The Annals of Applied Probability 13, 1119–1150.

Harrison, J.M., 2013. Brownian Models of Performance and Control. Cambridge University Press.

Harrison, J.M., Nguyen, V., 1993. Brownian models of multiclass queueing networks: current status and open problems. Queueing Systems 13, 5–40.

Harrison, J.M., Wein, L.M., 1989. Scheduling networks of queues: Heavy traffic analysis of a simple open network. Queueing Systems 5, 265–280.

Hassin, R., Puerto, J., Fernández, F.R., 2009. The use of relative priorities in optimizing the performance of a queueing system. European Journal of Operational Research 193, 476–483.

Herroelen, W., Leus, R., 2005. Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research 165, 289–306.

Hofri, M., Rosberg, Z., 1987. Packet delay under the golden ratio weighted TDM policy in a multiple-access channel. IEEE Transactions on Information Theory 33, 341–349.

Hooker, J.N., 2005. A hybrid method for planning and scheduling. Constraints 10, 385–401.

Huang, J., 2013. Patient Flow Management in Emergency Departments. Ph.D. thesis. National University of Singapore.

Jackson, J.R., 1963. Jobshop-like queueing systems. Management Science 10, 131–142.

Jaiswal, N.K., 1968. Priority Queues. Academic Press.

Jaiswal, N.K., 1982. Performance evaluation studies for time-sharing computer systems. Performance Evaluation 2, 223–236.

Jia, J., Heragu, S.S., 2009. Solving semi-open queuing networks. Operations research 57, 391–401.

Kaczynski, W.H., Leemis, L.M., Drew, J.H., 2011. Transient queueing analysis. INFORMS Journal on Computing To Appear.

Kang, W., Kelly, F.P., Lee, N.H., Williams, R.J., 2004. Fluid and Brownian approximations for an internet congestion control model, in: 43rd IEEE Conference on Decision and Control, pp. 3938–3943.

Kao, E.P.C., Narayanan, K.S., 1991. Analyses of an M/M/N queue with servers' vacations. European Journal of Operational Research 54, 256–266.

Katta, A.K., Sethuraman, J., 2005. A note on bandits with a twist. SIAM Journal on Discrete Mathematics 18, 110–113.

Kelly, F.P., Laws, C.N., 1993. Dynamic routing in open queueing networks: Brownian models, cut constraints and resource pooling. Queueing Systems 13, 47–86.

Kelly, F.P., Zachary, S., Ziedins, I. (Eds.), 1996. Stochastic Networks: Theory and Applications. Oxford University Press.

Khamisy, A., Altman, E., Sidi, M., 1992. Polling systems with synchronization constraints. Annals of Operations Research 35, 231–267.

Khintchine, A.Y., 1932. Mathematical theory of stationary queues. Matematicheskii Sbornik 39, 73–84. In Russian.

Kim, J., Ward, A.R., 2013. Dynamic scheduling of a GI/GI/1+ GI queue with multiple customer classes. Queueing Systems 75, 339–384.

Kitaev, M.Y., Rykov, V.V., 1995. Controlled Queueing Systems. 1st ed., CRC-Press.

Kleinrock, L., 1976. Queueing Systems, Volume 1 and 2. Wiley.

Klimov, G.P., 1975. Time-sharing service systems I. Theory of Probability and its Applications 19, 532–551. Translated from Russian.

Kreipl, S., Pinedo, M., 2004. Planning and scheduling in supply chains: an overview of issues in practice. Production and Operations Management 13, 77–92.

Kubiak, W., 2004. Fair sequences, in: Leung, J.Y.T. (Ed.), Handbook of Scheduling: Algorithms, Models and Performance Analysis. CRC Press. chapter 19, pp. 19–1–19–21.

Kumar, P.R., Meyn, S.P., 1995. Stability of queueing networks and scheduling policies. IEEE Transactions on Automatic Control 40, 251–260.

Kumar, S., Muthuraman, K., 2004. A numerical method for solving singular stochastic control problems. Operations Research 52, 563–582.

Kushner, H.J., Martins, L.F., 1996. Heavy traffic analysis of a controlled multiclass queueing network via weak convergence methods. SIAM Journal on Control and Optimization 34, 1781–1797.

van Leeuwaarden, J.S.H., Lefeber, E., Nazarathy, Y., Rooda, J.E., 2010. Model predictive control for the acquisition queue and related queueing networks, in: Proceedings of the 5th International Conference on Queueing Theory and Network Applications, ACM. pp. 193–200.

Lehoczky, J.P., 1996. Real-time queueing theory, in: Proceedings of the 17th IEEE Real-Time Systems Symposium, pp. 186–195.

Lenstra, J.K., Shmoys, D.B., 1995. Computing near-optimal schedules, in: Chrétienne, P., Coffman Jr., E.G., Lenstra, J.K., Liu, Z. (Eds.), Scheduling Theory and Its Applications. John Wiley & Sons Ltd. chapter 1, pp. 1–14.

Leon, V.J., Wu, S.D., Storer, R.H., 1994. Robustness measures and robust scheduling for job shop. IIE Transactions 26, 32–43.

Leung, J.Y.T. (Ed.), 2004. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press.

Levy, H., 1991. Binomial-gated service: A method for effective operation and optimization of polling systems. IEEE Transactions on Communications 39, 1341–1350.

Levy, H., Sidi, M., 1990. Polling systems: Applications, modeling, and optimization. IEEE Transactions on Communications 38, 150–1760.

Lu, S.C.H., Ramaswamy, D., Kumar, P.R., 1994. Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants. IEEE Transactions on Semiconductor Manufacturing 7, 374–388.

Maglaras, C., 1999. Dynamic scheduling in multiclass queueing networks: Stability under discrete-review policies. Queueing Systems 31, 171–206.

Maglaras, C., 2000. Discrete-review policies for scheduling stochastic networks: Trajectory tracking and fluid-scale asymptotic optimality. The Annals of Applied Probability 10, 897–929.

Maglaras, C., 2003. Continuous-review tracking policies for dynamic control of stochastic networks. Queueing Systems 43, 43–80.

Mahajan, A., Teneketzis, D., 2007. Multi-armed bandit problems, in: Hero III, A.O., Castañón, D.A., Cochran, D., Kastella, K. (Eds.), Foundations and Applications of Sensor Management. Springer, pp. 121–151.

Martins, L.F., Shreve, S.E., Soner, H.M., 1996. Heavy traffic convergence of a controlled, multiclass queueing system. SIAM Journal on Control and Optimization 34, 2133–2171.

Massey, W.A., Srinivasan, R., 1991. A heavy traffic analysis for semi-open networks. Performance Evaluation 13, 59–66.

Massey, W.A., Whitt, W., 1998. Uniform acceleration expansions for Markov chains with time-varying rates. Annals of Applied Probability 8, 1130–1155.

Meilijson, I., Yechiali, U., 1977. On optimal right-of-way policies at a single-server station when insertion of idle times is permitted. Stochastic Processes and Their Applications 6, 25–32.

Mercier, L., Van Hentenryck, P., 2008. Amsaa: A multistep anticipatory algorithm for online stochastic combinatorial optimization, in: Proceedings of the 5th International Conference on Integration of AI and OR techniques in constraint programming for combinatorial optimization problems (CPAIOR'08), pp. 173–187.

Meyn, S., 1997. Stability and optimization of queueing networks and their fluid models. LECTURES IN APPLIED MATHEMATICS-AMERICAN MATHEMATICAL SOCIETY 33, 175–200.

Meyn, S.P., 2001. Sequencing and routing in multiclass queueing networks. Part I: Feedback regulation. SIAM Journal on Control and Optimization 40, 741–776.

Meyn, S.P., 2008. Control Techniques for Complex Networks. Cambridge University Press.

Moallemi, C.C., Kumar, S., Van Roy, B., 2008. Approximate and data-driven dynamic programming for queueing networks. Working paper, Available at https://moallemi.com/ciamac/papers/adp-queueing-2008.pdf.

Monahan, G.E., 1982. A survey of partially observable Markov decision processes: Theory, models, and algorithms. Management Science 28, 1–16.

Moustafa, M.S., 1987. Scheduling in a multi-class network. Ph.D. thesis. North Carolina State University.

Moustafa, M.S., 1996. Optimal assignment policy of a single server attended by two queues. Applied Mathematics and Computation 80, 245–255.

Myers, K., Berry, P., Blythe, J., Conley, K., Gervasio, M., McGuinness, D., Morley, D., A., P., Pollack, M., Tambe, M., 2007. An intelligent personal assistant for task and time management. AI Magazine 28, 47–61.

Nash, P., Weber, R.R., 1982. Sequential open-loop scheduling strategies, in: Dempster, M.A.H., K., L.J., Rinnooy Kan, A.H.G. (Eds.), Deterministic and Stochastic Scheduling: proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems. D. Reidel Publishing Company, pp. 385–397.

Nazarathy, Y., Weiss, G., 2009. Near optimal control of queueing networks over a finite time horizon. Annals of Operations Research 170, 233–249.

Nazarathy, Y., Weiss, G., 2010. A fluid approach to large volume job shop scheduling. Journal of Scheduling 13, 509–529.

Niño-Mora, J., 2007. Dynamic priority allocation via restless bandit marginal productivity indices. TOP 15, 161–198.

Nuyens, M., Wierman, A., Zwart, A.P., 2008. Preventing large sojourn times with SMART scheduling. Operations Research 56, 88–101.

Oliver, R.M., Pestalozzi, G., 1965. On a problem of optimum priority classification. Journal of the Society for Industrial and Applied Mathematics 13, 890–901.

Ouelhadj, D., Petrovic, S., 2009. A survey of dynamic scheduling in manufacturing systems. Journal of Scheduling 12, 417–431.

Pardo, M.J., de la Fuente, D., 2007. Optimizing a priority-discipline queueing model using fuzzy set theory. Computers and Mathematics with Applications 54, 267–281.

Paschalidis, I.C., Su, C., Caramanis, M.C., 2003. Target-pursuing policies for open multiclass queueing networks, in: Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM'03), pp. 196–206.

Paschalidis, I.C., Su, C., Caramanis, M.C., 2004. Target-pursuing scheduling and routing policies for multiclass queueing networks. IEEE Transactions on Automatic Control 49, 1709–1722.

Patrick, J., Puterman, M.L., Queyranne, M., 2008. Dynamic multipriority patient scheduling for a diagnostic resource. Operations Research 56, 1507–1525.

Pegden, C.D., Rosenshine, M., 1990. Scheduling arrivals to queues. Computers and Operations Research 17, 343–348.

Phipps Jr., T.E., 1956. Machine repair as a priority waiting-line problem. Operations Research 4, 76–85.

Pinedo, M.L., 2003. Scheduling: Theory, Algorithms, and Systems. 2 ed., Prentice-Hall.

Pinedo, M.L., 2009. Planning and Scheduling in Manufacturing and Services. Springer.

Plambeck, E.L., Fu, B.R., Robinson, S.M., Suri, R., 1996. Sample-path optimization of convex stochastic performance functions. Mathematical Programming 75, 137–176.

Pollaczek, F., 1932. Lösung eines geometrischen wahrscheinlichkeitsproblems. Mathematische Zeitschrift 35, 230–278. In German.

Posner, M., Bernholtz, B., 1968. Closed finite queuing networks with time lags and with several classes of units. Operations Research 16, 977–985.

Powell, W.B., 2010. Approximate dynamic programming I: Modeling, in: Cochran, J.J., Cox, L.A., Keskinocak, P., Kharoufeh, J.P., Smith, J.C. (Eds.), Wiley Encyclopedia of Operations Research and Management Science. John Wiley and Sons, Inc.

Powell, W.B., 2012. Perspectives of approximate dynamic programming. Annals of Operations Research , 1–38.

Pruhs, K., 2007. Competitive online scheduling for server systems. ACM SIGMETRICS Performance Evaluation Review 34, 52–58.

Puterman, M.L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc.

Radlinski, F., Kleinberg, R., Thorsten, J., 2005. Learning diverse rankings with multi-armed bandits, in: Proceedings of Twenty Fifth International Conference on Machine Learning (ICML'08), pp. 1355–1361.

Raviv, T., 2003. Fluid Approximation and Other Methods for Hard Combinatorial Optimization Problems. Ph.D. thesis. Technion-Israel Institute of Technology, Haifa, Israel.

Reiman, M.I., Wein, L.M., 1998. Dynamic scheduling of a two-class queue with setups. Operations Research , 532–547.

Reiman, M.I., Wein, L.M., 1999. Heavy traffic analysis of polling systems in tandem. Operations Research 47, 524–534.

Righter, R., 1994. Scheduling, in: Shaked, M., Shanthikumar, J.G. (Eds.), Stochastic Orders and Their Applications. Academic Press, pp. 381–432.

Robinson, D., 1978. Optimization of priority queues – a semi-Markov decision chain approach. Management Science 24, 545–553.

Robinson, S.M., 1996. Analysis of sample-path optimization. Mathematics of Operations Research 21, 513–528.

Rosa-Hatko, W., Gunn, E.A., 1997. Queues with switchover – a review and critique. Annals of Operations Research 69, 299–322.

Ross, S.M., 2003. Introduction to Probability Models. Academic Press. chapter 6 – Continuous-Time Markov Chains. pp. 349–399.

Rothkopf, M.H., 1966. Scheduling with random service times. Management Science 12, 707–713.

Ruschitzka, M., Fabry, R.S., 1977. A unifying approach to scheduling. Communications of the ACM 20, 469–477.

Schrage, L., 1968. A proof of the optimality of the shortest remaining processing time discipline. Operations Research 16, 687–690.

Schrage, L., 1970. An alternative proof of a conservation law for the queue G/G/1. Operations Research 18, 185–187.

Schuurman, P., Woeginger, G.J., 1999. Polynomial time approximation algorithms for machine scheduling: Ten open problems. Journal of Scheduling 2, 203–213.

Sennott, L.I., 1999. Stochastic Dynamic Programming and the Control of Queueing Systems. John Wiley & Sons, Inc.

Sha, L., Abdelzaher, T., Årzén, K.E., Cervin, A., Baker, T., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., Mok, A.K., 2004. Real time scheduling theory: A historical perspective. Real-time systems 28, 101–155.

Shanthikumar, J.G., 1982. On reducing time spent in M/G/1 systems. European Journal of Operational Research 9, 286–294.

Shanthikumar, J.G., Ding, S., Zhang, M.T., 2007. Queueing theory for semiconductor manufacturing systems: A survey and open problems. IEEE Transactions on Automation Science and Engineering 4, 513–522.

Sharafali, M., Co, H.C., Goh, M., 2004. Production scheduling in a flexible manufacturing system under random demand. European Journal of Operational Research 158, 89–102.

Skaf, J., Boyd, S.P., 2010. Design of affine controllers via convex optimization. IEEE Transactions on Automatic Control 55, 2476–2487.

Smith, D.R., 1978. A new proof of the optimality of the shortest remaining processing time discipline. Operations Research 26, 197–199.

Smith, W.E., 1956. Various optimizers for single-stage production. Naval Research Logistics Quarterly 3, 59–66.

Sotskov, Y.N., Sotskova, N.Y., Lai, T.C., Werner, F., 2010. Scheduling Under Uncertainty: Theory and Algorithms. Belorussian Science.

Stidham, S.J., 1985. Optimal control of admission to a queueing system. IEEE Transactions on Automatic Control AC-30, 705–713.

Stidham, S.J., 2002. Analysis, design, and control of queueing systems. Operations Research 50, 197–216.

Stidham, S.J., 2009. Optimal Design of Queueing Systems. CRC-Press.

Stidham, S.J., Weber, R., 1993. A survey of Markov decision models for control of networks of queues. Queueing Systems 13, 291–314.

Stolyar, A.L., 2004. Maxweight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic. The Annals of Applied Probability 14, 1–53.

Streeter, M., 2007. Using Online Algorithms to Solve NP-hard Problems More Efficiently in Practice. Ph.D. thesis. Carnegie Mellon University, Pittsburgh, PA.

Su, H., 2006. Robust fluid control of multiclass queueing networks. Ph.D. thesis. Massachusetts Institute of Technology.

Suresh, V., Chaudhuri, D., 1993. Dynamic scheduling – a survey of research. International Journal of Production Economics 32, 53–63.

Tadj, L., Choudhury, G., 2005. Optimal design and control of queues. Sociedad de Estadìstica e Investigación Operativa, Top 13, 359–412.

Takagi, H., 1986. Analysis of Polling Systems. MIT Press.

Takagi, H., 1988. Queueing analysis of polling models. ACM Computing Surveys 20, 5–28.

Tassiulas, L., Bhattacharya, P.P., 2000. Allocation of interdependent resources for maximal throughput. Stochastic Models 16, 27–48.

Tassiulas, L., Ephremides, A., 1992. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE Transactions on Automatic Control 37, 1936–1948.

Teh, Y.C., 2000. Dynamic scheduling for queueing networks derived from discrete-review policies, in: McDonald, D.R., Turner, S.R.E. (Eds.), Analysis of Communication Networks: Call Centres, Traffic, and Performance. AMS Bookstore, pp. 73–95.

Terekhov, D., 2013. Integrating Combinatorial Scheduling with Inventory Management and Queueing Theory. Ph.D. thesis. Department of Mechanical and Industrial Engineering, University of Toronto.

Terekhov, D., Beck, J.C., Brown, K.N., 2009. A constraint programming approach for solving a queueing design and control problem. INFORMS Journal on Computing 21, 549–561.

Terekhov, D., Down, D.G., Beck, J.C., 2012a. Stability of a Polling System with a Flow-Shop Server. Technical Report MIE-OR-TR2012-01. Department of Mechanical and Industrial Engineering, University of Toronto. Available from http://www.mie.utoronto.ca/labs/ORTechReps/.

Terekhov, D., Tran, T.T., Down, D.G., Beck, J.C., 2012b. Long-run stability in dynamic scheduling problems, in: Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12), pp. 261–269.

Terekhov, D., Tran, T.T., Down, D.G., Beck, J.C., 2014. Integrating queueing theory and scheduling for dynamic scheduling problems. Journal of Artificial Intelligence Research 50, 535–572.

Tian, N., Zhang, Z.G., 2006. Vacation Queueing Models: Theory and Applications. Springer.

Tran, T.T., 2011. Using Queueing Analysis to Guide Combinatorial Scheduling in Dynamic Environments. Master's thesis. Department of Mechanical and Industrial Engineering, University of Toronto.

Tran, T.T., Terekhov, D., Down, D.G., Beck, J.C., 2013. Hybrid queueing theory and scheduling models for dynamic environments with sequence-dependent setup times, in: Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13), pp. 215–223.

Van Der Wal, J., Yechiali, U., 2003. Dynamic visit-order rules for batch-service polling. Probability in the Engineering and Informational Sciences 17, 351–367.

Van Hentenryck, P., Bent, R., 2006. Online Stochastic Combinatorial Optimization. MIT Press.

Varaiya, P., Walrand, J., Buyukkoc, C., 1985. Extensions of the multiarmed bandit problem: the discounted case. IEEE Transactions on Automatic Control 30, 426–439.

Veatch, M.H., 2003. Using fluid solutions in dynamic scheduling, in: Gershwin, S.B., Dallery, Y., Papadopoulos, C.T., M., S.J. (Eds.), Analysis and modeling of manufacturing systems. Springer. volume 60 of *International Series in Operations Research and Management Science*. chapter 16, pp. 399–426.

Veatch, M.H., 2009. Approximate dynamic programming for networks: Fluid models and constraint reduction. Working paper, Available at http://faculty.gordon.edu/ns/mc/mike_veatch/documents/reduction1.pdf.

Veatch, M.H., 2010. Approximate linear programming for networks: Average cost bounds. Technical Report. Working paper, Gordon College, Dept. of Math. Available at http://faculty. gordon. edu/ns/mc/mike-veatch.

Veatch, M.H., Walker, N., 2008. Approximate linear programming for network control: Column generation and subproblems. Working paper, Available at http://faculty.gordon.edu/ns/mc/mike_veatch/documents/colgen.pdf.

Veatch, M.H., Wein, L.M., 1992. Monotone control of queueing networks. Queueing Systems 12, 391–408.

Vermorel, J., Mohri, M., 2005. Multi-armed bandit algorithms and empirical evaluation, in: Proceedings of the 16th European Conference on Machine Learning (ECML'05), pp. 437–448.

Vishnevskii, V.M., Semenova, O.V., 2006. Mathematical methods to study the polling systems. Automation and Remote Control 67, 173–220.

Weber, R.R., Stidham, S.J., 1987. Optimal control of service rates in networks of queues. Advances in Applied Probability 19, 202–218.

Weber, R.R., Weiss, G., 1990. On an index policy for restless bandits. Journal of Applied Probability , 637–648.

Wein, L.M., 1990. Scheduling networks of queues: Heavy traffic analysis of a two-station network with controllable inputs. Operations Research 38, 1065–1078.

Wein, L.M., 1994. Scheduling networks of queues: Heavy traffic analysis of a bi-criteria problem, in: Yao, D.D. (Ed.), Stochastic Modeling and Analysis of Manufacturing Systems. Springer. Springer Series in Operations Research, pp. 281–323.

Wein, L.M., Chevalier, P.B., 1992. A broader view of the job-shop scheduling problem. Management Science 38, 1018–1033.

Weiss, G., 1988. Branching bandit processes. Probability in the Engineering and Informational Sciences 2, 269–278.

Whittle, P., 1988. Restless bandits: Activity allocation in a changing world. Journal of Applied Probability 25A, 287–298.

Wierman, A., Harchol-Balter, M., Osogami, T., 2005. Nearly insensitive bounds on SMART scheduling. ACM SIGMETRICS Performance Evaluation Review 33, 205–216.

Wierman, A., Winands, E., Boxma, O., 2007. Scheduling in polling systems. Performance Evaluation 64, 1009–1028.

van Wijk, A.C.C., Adan, I.J.B.F., Boxma, O.J., Wierman, A., 2012. Fairness and efficiency for polling models with the κ-gated service discipline. Performance Evaluation 69, 274–288.

Williams, R.J., 1996. On the approximation of queueing networks in heavy traffic, in: Kelly, F.P., Zachary, S., Ziedins, I. (Eds.), Stochastic Networks: Theory and Applications. Oxford University Press. chapter 3, pp. 35–56.

Williams, R.J., 1998. Diffusion approximations for open multiclass queueing networks: sufficient conditions involving state space collapse. Queueing Systems 30, 27–88.

Winands, E.M.M., 2007. Polling, production and priorities. Ph.D. thesis. Technische Universiteit Eindhoven.

Wolff, R.W., 1989. Stochastic modeling and the theory of queues. Prentice Hall.

Wu, C.W., Brown, K.N., Beck, J.C., 2009. Scheduling with uncertain durations: Modeling β-robust scheduling with constraints. Computers and Operations Research 36, 2348–2356.

Wu, K., 2014. Classification of queueing models for a workstation with interruptions: a review. International Journal of Production Research 52, 902–917.

Yechiali, U., 1991. Optimal dynamic control of polling systems. Queueing, Performance and Control in ATM (ITC-13) , 205–217.

Yechiali, U., 1993. Analysis and control of polling systems. Performance Evaluation of Computer and Communication Systems , 630–650.

Yeow, W.L., Tham, C.K., Wong, W.C., 2006. Hard constrained semi-Markov decision processes, in: Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI'06), pp. 549–555.

Zonderland, M.E., Boucherie, R.J., Litvak, N., Vleggeert-Lankamp, C.L.A.M., 2010. Planning and scheduling of semi-urgent surgeries. Health Care Management Science 13, 256–267.