The Dissertation Committee for Burak Buke
certifies that this is the approved version of the following dissertation:

Optimal Draining of Fluid Networks with Parameter Uncertainty

Committee:

John J. Hasenbein, Supervisor

David P. Morton, Supervisor

Elmira Popova

Erhan Kutanoglu

Sanjay Shakkottai

Optimal Draining of Fluid Networks with Parameter Uncertainty

by

Burak Buke, B.S., M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2007

In the memory of my father...

Acknowledgments

I would like to start with expressing my gratitude to my advisors Dr. John Hasenbein and Dr. David Morton. I would not be able to put this work together without their invaluable support. I especially want to thank them for supporting, motivating and encouraging me. Not only they guided me in various ways in my Ph.D. journey, but they also taught me how to be a good academic. I would also like to thank my committee members Dr. Popova, Dr. Kutanoglu and Dr. Shakkottai. Surely, their help during my graduate studies extends beyond this dissertation. I would like to thank Dr. Nuri Uman of Koc University and Dr. Aytul Ercil of Sabanci University for encouraging me to pursue an academic career.

I have really enjoyed the time I spent in Austin. At this point, I also take this opportunity to thank my wonderful friends in Austin. I want to thank Fehmi for supporting me throughout my graduate life. I also would like to thank Ali, Murat, Guray, Utku, Erol, Emrah Tanriverdi, Ulas, Ferhat, Emrah Zarifoglu, Armagan, Amit, Kranthi, Vishv Jeet, Mukremin and numerous others. I thank everyone who has been a part of my life in Austin.

Last, but not the least, I would like to thank all the people in my family for their endless support. Without them believing in me, it would be impossible to come up with this work.

Optimal Draining of Fluid Networks with Parameter Uncertainty

Publication No. ____________________

Burak Buke, Ph.D.
The University of Texas at Austin, 2007

Supervisors: John J. Hasenbein
David P. Morton

Fluid networks are useful tools for analyzing complex manufacturing environments especially in semiconductor wafer fabrication. The makespan of a fluid network is defined as the time to drain the system, when there is fluid present in the buffers initially. Based on this definition, the question of determining the allocation of resources so as to minimize the makespan of a fluid network is known as the makespan problem. In the deterministic version of the makespan problem, it is assumed that the parameters of the system, such as incoming rates, service rates and initial inventory, are known deterministically.

The deterministic version of the makespan problem for reentrant lines and multiclass fluid networks has been investigated in the literature and an analytical solution for the problem is well-known. In this work, we provide another formulation for the deterministic makespan problem and prove that the problem can be solved for each station separately. Optimal solutions for the deterministic makespan problem have been used as a guide to develop heuristics methods to solve makespan scheduling problem in the job-shop context in the literature. This provides one motivation for further investigation of the fluid makespan problem.

In this work our main focus is solving the makespan problem when the problem parameters are uncertain. This uncertainty may be caused by various factors such as the unpredictability of the arrival process or randomness in machine availability due to failures. In the presence of parameter uncertainty, the decision maker's goal is to optimally allocate the capacity in order to minimize the expected value of the makespan. We assume that the decision maker has distributional information about the parameters at the time of decision making.

We consider two decision making schemes. In the first scheme, the controller sets the allocations before observing the parameters. After the initial allocations are set, they cannot be changed. In the second scheme, the controller is allowed a recourse action after a data collection process. It is shown that in terms of obtaining the optimal control, both schemes differ considerably from the deterministic version of the problem. We formulate both schemes using stochastic programming techniques. The first scheme is easier to analyze since the resulting model is convex. Unfortunately, under the second decision scheme, the objective function is non-convex. We develop a branch-and-bound methodology to solve the resulting stochastic non-convex program. Finally, we identify some special cases where the stochastic problem is analytically solvable.

This work uses stochastic programming techniques to formulate and solve a problem arising in queueing networks. Stochastic programming and queueing systems are two major areas of Operations Research that deal with decision making under uncertainty. To the best of our knowledge, this dissertation is one of the first works that brings these two major areas together.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Dealing with problems arising in the real world has always been a hard task. Recent developments in complex manufacturing environments and communication systems have introduced new and challenging problems for researchers. Some of the most challenging of these problems arise in the semiconductor wafer fabrication industry. The production of a wafer includes hundreds of steps. More importantly, the production is highly reentrant, i.e., the jobs visit some stations more than once during the production process. The reason for this situation is the high machine costs. Moreover, in general the cost of a wafer fab should be paid off within three years, which makes the efficient design and control of the production system absolutely necessary.

In optimization of the process, the most important issue is how the processing network is modeled. In recent years, there has been considerable effort to develop mathematical models to incorporate the important characteristics of complex processing networks. One of the most commonly used models that addresses many of the issues arising in processing networks is a so-called *multiclass queueing network*. A multiclass queueing network provides a framework to model the reentrant structure of the processing network. Despite being a fairly realistic representation of real systems, multiclass queueing networks are much more difficult to analyze than the traditional queueing network models introduced by Jackson [24] and Kelly [27].

To obtain a tractable model for real problems, methods have been proposed to approximate multiclass queueing networks. One approximation that allows us to perform efficient analysis is known as the *fluid model.* The fluid model can be viewed as a deterministic counterpart to the original queueing network. In a fluid network, instead of discrete jobs, a continuous fluid is assumed to flow through the processing network. In a multiclass fluid network, fluid arrives to various buffer in a network at known, constant rates. Fluid is then processed at a station at a given rate and then either routed to another station for processing, or it leaves the network. A theoretical justification for why this model is a valid approximation of the original discrete queueing model may be found in Dai [12] and Chen and Yao [11].

A standard optimization problem in fluid networks is to drain the network in the least amount of time, given an initial fluid inventory. This problem is sometimes referred as the *fluid makespan problem* or the *clearing time problem.* A closely related problem is to drain the network with the lowest cost, where the cost is some function of the fluid levels in the network. This problem is known as the *fluid holding cost problem.* When the parameters of the system are known, the makespan problem is a relatively simple optimization problem, which only requires the inversion of the fluid routing matrix. The holding cost problem is much more difficult in general, and falls into the class of separated continuous linear programs (see, e.g., Anderson and Nash [1]).

In this work, we concentrate only on the fluid model. Our model is an extension of the fluid model described above in that we allow some of the parameters, specifically the fluid arrival and processing rates, and the initial inventory to be random vectors. We refer to such a network as a *stochastic fluid model*. Our goal is to minimize the expected value of the makespan when parameters are not known deterministically. This version of the makespan problem with uncertain parameters is referred to as the *stochastic makespan problem*. We focus on the makespan objective due to its relative simplicity, although some results also apply to more general objective functions.

We view our models as useful approximations of reality in systems where at least the following characteristics are present: (1) the dynamical aspects of the system are well-approximated by a multiclass fluid model, in particular the possible discrete nature of the system and small time scale stochastic fluctuations are well represented by a deterministic, continuous model; (2) the stochastic behavior of some structural parameters of the system are dominant in terms of system behavior; and, (3) the decision maker is constrained in the sense that some training or allocation decisions must be made before the stochastic structural parameters can be measured. Systems in a number of different application areas do have these characteristics and we mention just a few. In a recent paper, Harrison and Zeevi [21] present a compelling argument for using a model of a similar nature in call center applications. In particular, in their model incoming calls are approximated on a local time scale by a deterministic fluid process. However, over longer time scales, they assume that the incoming call rates have some stochastic variability which is the dominant random factor. Finally, they posit that call center staffing decisions must be made *before* the incoming call rates are known. Thus, their modeling framework for call centers coincides with our modeling regime.

In semiconductor wafer manufacturing, the dynamics of the manufacturing process can often be well-approximated by a multiclass fluid model when there is a high production volume in the wafer fab. The dominant uncertainties in a wafer fab are usually in terms of demand rates (i.e., lot arrival rates) and the availability of critical equipment, due to unscheduled downtimes. In some cases, machine purchases, reticle availability, and setups constrain the local time scale decisions of machines allocated to different products. Hence, this application domain provides another motivation for our model. These considerations provide the motivation to study the stochastic version of the makespan problem for fluid networks which was analyzed in Weiss [37] and Dai and Weiss [13].

We consider two different decision making schemes. In the first scheme, just before time 0, the decision maker must choose a set of "allocation percentages" v_k, which determine what percentage of a server's capacity will be devoted to class k fluid (assuming there is a sufficient amount of fluid to be worked on). These percentages are then fixed once and for all. At time 0, when the system begins operation, a realization of the stochastic parameters is revealed, and the system then operates under that realization and the allocation percentages v_k which were chosen. In the second decision scheme, the controller has more freedom on setting the allocations. As in the first scheme, the initial allocations v_ks are set just before time 0. However, now the controller has the opportunity to change her decision after some possibly random time T. In both cases, the controller's goal is to choose the initial allocations v_k in a manner that minimizes the expected draining time of the system.

A number of different stochastic fluid models have been introduced in the literature. These models differ from ours in terms of the decision structure. In previous work, one is usually allowed to make allocation or admission decisions as soon as a change in the system parameters is observed, and in this sense, the decision structure is that of real-time control. In contrast, the decision structure of our models is that of a time-static stochastic program.

In our first decision structure, the controller must commit to a decision ahead of time and then live with the consequences of that decision no matter the realization of the stochastic parameters. This structure is reasonable when the controller should make a decision concerning the design of the system. For example, the number of dedicated servers (e.g., the number of trained personnel) to accomplish certain tasks should be decided before the system starts running and it may be too costly, or logistically impossible, to change this decision after the system parameters are observed. Another example, where it is not possible to modify the decision after realizing the parameters is signal control for heavy traffic in urban areas. The controller must decide ahead of time on the duration of red and green lights at each intersection without observing the actual flow in the system. In this case, induction sensors at intersections do not provide sufficient real-time information on the flow at every intersection.

In our second decision structure, the controller learns system parameters, and change the initial allocations accordingly after some time T. This delay may be due to several reasons. First, the controller may not be able to observe the system parameters immediately. In real life, it is only possible to obtain reasonable estimates for the system parameters after a data collection process. Hence, the controller has to wait to obtain the results of the data collection process. Secondly, even if the data collection process can be neglected, the controller may need to change some structural changes (e.g., train personnel to shift them from one task to the other). The controller's goal is to make the optimal decisions till the allocations can be changed.

Perhaps the model closest to ours in spirit is in the aforementioned paper Harrison and Zeevi [21]. Harrison and Zeevi [21] studied a more general server structure than ours, since their network has flexible servers, i.e., a fluid class may be served by more than one station in the network. However, their network structure is simpler, since they only consider "one pass" networks in which fluid visits only one server and then departs the network. In the context of the makespan problem, we do not consider flexible servers; however, taking the possible application areas into account, we consider a reentrant structure for the network. Also in contrast to [21], we focus on the structural aspects of the stochastic programming problem which arises. As in [21], Atlason et al. [3]

optimize staffing levels at a call center but they also address the combinatorial problem of constructing employee shifts while using a simulation model to estimate the center's performance. Gürkan [19] selects constrained buffer sizes to optimize throughput in a fluid tandem queueing network with random machine failures, also using simulation to estimate steady-state throughput. The recent papers of Iyengar and Zeevi [23] and Whitt [39] also examine queueing models with decision structures that closely resemble ours.

The parameter uncertainty issue is also addressed in the literature for various other applications than call centers. There is a large body of work on models related to the classical Anick-Mitra-Sondhi [2] stochastic fluid model. In those models, generally speaking, service rates are deterministic and arrival rates vary according to an underlying Markov chain (i.e., the arrival process is *Markov modulated*). The controller's job is usually to determine which fluid classes to serve at any given time and how much of each fluid type to admit to the system in order to minimize a cost function. For modeling of manufacturing systems, both the incoming fluid rate and the processing rates at a server may be allowed to vary according to some stochastic process. Again, in those models the controller may be allowed to control both admissions to the system and the servers' time allocations. For different approaches to these problems see, for example, Bäurle [5, 6], Sun et al. [35], and Gürkan et al. [20]. Overviews of stochastic fluid models used in the manufacturing and telecommunications application appear in Kulkarni [28] and Sethi et al. [33].

The makespan and holding cost problems for deterministic fluid networks have been studied in a number of papers. A short list includes Avram et al. [4], Billings [9], Chen and Yao [11], Pullan [31, 32] and Weiss [37, 38]. The optimal controls of the fluid optimization problem can be used to construct efficient heuristics to solve the discrete counterpart of the problem. Bertsimas and Sethuraman [8], and Dai and Weiss [13] provide asymptotically optimal heuristics for the makespan problem in queueing networks based on the fluid approximation. Similarly, Bertsimas et al. [7] provide an asymptotically optimal heuristic for the holding cost problem in a queueing system.

The dissertation is organized as follows. In Chapter 2, we introduce the deterministic version of the makespan problem. Even though the deterministic problem is analytically solvable, we introduce a mathematical programming formulation of the problem that helps us in analyzing the stochastic version of the problem.

In Chapter 3, we analyze our first decision structure. We provide some counterexamples to show how the stochastic problem differs from the deterministic version. We state conditions for the well-posedness of the problem and formulate the problem using a stochastic programming model. Finally, we discuss the methodologies that can be used to solve the problem.

In Chapter 4, the second decision structure is analyzed, in which a recourse action is allowed. We formulate the problem under this decision structure, which turns out to be a stochastic non-convex programming model. We then develop a branch-and-bound methodology, which is guaranteed to give the optimal expected makespan after a finite number of iterations.

Finally, in Chapter 5, we identify some special cases where the optimal solution can be characterized analytically. Then, we outline our contributions in Chapter 6 and give further directions for research.

Chapter 2

Makespan Problem

In this chapter, we give a formal definition of the makespan problem in fluid networks. In Section 2.1, we introduce the notation that is used throughout this work, and using that notation, state the equations governing the fluid process. Then, we define the controls used to solve the makespan problem. If all the parameters of the system are known deterministically, the makespan problem is a well-solved problem. In Section 2.2, we state the solution of the deterministic makespan problem as given in Chen and Yao [11]. Then we give an optimization model that yields the optimum solution of the deterministic makespan problem. Even though the deterministic problem can be solved analytically, this model enables us to analyze the stochastic case more effectively. This model also allows us to observe the separability property of the deterministic makespan problem.

2.1 Modeling and Notation

In this work, we consider a fluid model where fluid flows through a system consisting of stations indexed by $j \in J$ and classes of fluid indexed by $k \in K$. We envision each class of fluid being stored in a buffer, which we refer as buffer k. We assume $|J| \leq |K|$ and that each class k is served by a unique station $\sigma_k \in J$. On the other hand, station j drains a set of buffers denoted by $\mathcal{C}_j$, where $\mathcal{C}_j = \{k|\sigma_k = j\}$. The system starts with an initial inventory a_k in each buffer k. Fluid arrives to buffer k from outside the system at rate α_k. If station σ_k allocates all its effort to buffer k, it takes m_k units of time to drain one unit of fluid from buffer k. When subscripts are omitted a, α and m denote the vector forms of the above parameters. All vectors are assumed to be column vectors. After the fluid leaves buffer k some portions of the fluid is routed to the buffers in the system and the remaining portion leaves the system. The proportion of fluid that is routed to buffer l from buffer k is denoted p_{kl}. The $|K| \times |K|$ matrix P, with elements p_{kl}, is called the routing matrix. In this work matrices are denoted by upper case letters and to denote the k^{th} row of a matrix, the superscript k notation is used. To avoid confusion, when a superscript is used as a power operator, the matrix is written in parentheses. We use I to denote an appropriately-dimensioned identity matrix and e to denote vector of all 1s.

The system described above is called a multiclass fluid network. An example of a multiclass fluid network with 2 stations and 4 buffers is given in Figure 2.1. A multiclass fluid process is given by $(Z(t), T(t))$ for $t \geq 0$. In this notation $Z(t)$ and $T(t)$ are $|K|$ dimensional. $T_k(t)$ is the total amount of effort (in units of time) spent to drain buffer k up to time t and $Z_k(t)$ gives the amount of fluid in buffer k at time t. In the above notation, $Z(0) = a$. We also define the service-time matrix, $M = \text{diag}(m)$. Then the dynamical

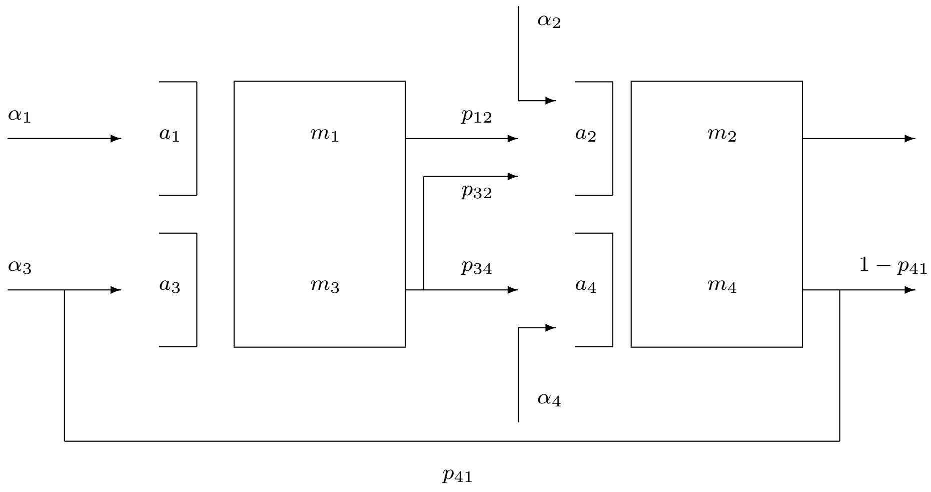

Figure 2.1: A Multiclass Network with Two Stations and Four Buffers

equations governing the fluid process are, for all $t \geq 0$:

$$Z(t) = a + \alpha t - (I - P')M^{-1}T(t) \tag{2.1a}$$

$$\sum_{k \in \mathcal{C}_j} T_k(t) \leq t, \quad j \in J \tag{2.1b}$$

$$Z(t) \geq 0 \tag{2.1c}$$

$$T_k(\cdot) \text{ nondecreasing, } T_k(0) = 0 \text{ for each } k \in K. \tag{2.1d}$$

A control policy is defined by a set of functions $\{T_k(t), k \in K\}$ on $[0, \infty)$. Once a policy $T(\cdot)$ is specified, then $Z(\cdot)$ is determined by (2.1a). If the resulting $Z(\cdot)$ satisfies (2.1c) and $T(\cdot)$ satisfies (2.1b) and (2.1d) then the solution is a feasible fluid solution.

Now, suppose we define $v_k(t) = \dot{T}_k(t)$ for all k and all $t \geq 0$ for which the derivative exists. It can be shown that $T(\cdot)$ is absolutely continuous and so its derivatives exist a.e. Then the functions $v_k(\cdot)$ provide an equivalent way to specify the control. One can interpret $v_k(t)$ as the instantaneous percentage of effort at station σ_k devoted to draining buffer k. In the stochastic setting we will find it easier to specify a control policy via $v(\cdot) \equiv (v_k(\cdot))$.

A policy is said to be stationary if these percentage allocations are not functions of time, i.e., if there is fluid in the system, the workload for buffer k is drained with rate v_k, and only the incoming workload is drained when buffer k is empty. Clearly, we have

$$\sum_{k \in \mathcal{C}_j} v_k \leq 1, \forall j \in J. \tag{2.2}$$

The makespan of a fluid network is the time that the network is actually drained, i.e., the minimum t such that $Z(t) = 0$. In this work, we analyze the problem of minimizing the makespan of a given fluid network by deciding on the allocation of effort at each station. Studying the makespan of fluid networks only makes sense if the networks are *open*, i.e., all fluid in the system will eventually leave the system. This notion makes more intuitive sense for queueing networks with discrete customers, but it turns out that it is necessary to adopt the same notion for fluid networks. To simplify the notation throughout the paper we define:

$$Q = I + P' + (P')^2 + \cdots .$$

An open fluid network is one for which the sum above converges. In that case,

Q is well defined and its expression reduces to $Q = (I - P')^{-1}$. We refer the reader to [12] for a detailed analysis of open fluid networks.

To ensure that the network can be drained, we need to enforce further conditions on the network parameters. The effective arrival rate of class k is $Q^k \alpha$, and so the amount of work which arrives to the system in unit time, destined for buffer k, is given by $Q^k \alpha m_k$. To be able to eventually drain the system, each station j must have enough capacity to process the total work which arrives to the system and is destined for the buffers in $\mathcal{C}_j$. That is, the following inequalities must hold:

$$\textstyle\sum_{k \in \mathcal{C}_j} Q^k \alpha m_k \leq 1, \quad \forall j \in J. \tag{2.3}$$

The conditions given in (2.3) are called *the usual traffic conditions* in the literature. When the inequality holds strictly, we say that *the strict usual traffic conditions hold.*

In [37], the makespan problem is examined for reentrant lines in a deterministic setting, i.e., the parameters of the system are known deterministically at the time of decision making. A reentrant line is a special type of multiclass queueing network, where only the first buffer receives exogenous input and proportional routing is not allowed in the system. In [37] it is shown that if the strict usual traffic conditions hold, there exists a policy which drains a reentrant line in finite time. A similar result holds for multiclass fluid networks, is shown in [11].

The main focus of this work is how to minimize the expected makespan of a multiclass fluid system over stationary policies, when the parameters a, α and m are not known deterministically at the time of decision making. The stationarity assumption is crucial, since it can be shown that, in some cases, all stationary policies are suboptimal when the optimization includes time-varying policies. However, if the stationarity assumption is removed, the problem becomes much more complex to analyze. We denote the sample space of the random variables by Ω and $\omega \in \Omega$ denotes a sample point in the sample space. A random variable x is generally denoted by $\tilde{x}$, and we use $x(\omega)$ to denote a realization of this random variable under $\omega \in \Omega$.

2.2 Deterministic Makespan Problem

Under a given policy, the makespan of a fluid network is defined as the time that the system reaches the empty state. We seek a policy that drains the system in minimal time. In the deterministic version of the problem, we assume that the parameters for the system, i.e., a, α and m are known deterministically at the time of decision making. This problem can be formulated as follows:

$$\begin{aligned} t^* = \min \quad & \int_0^\infty 1_{\{e'Z(t)>0\}} dt \\ \text{s.t.} \quad & (2.1a)\text{-}(2.1d). \end{aligned}$$

Here, $1_{\{e'Z(t)>0\}}$ is the indicator function, which takes value 1 if there is fluid in the network at time t and 0 otherwise. Taking the integral over time, we obtain the total time that there is fluid in the system.

The solution to the deterministic problem of minimizing makespan for general multiclass fluid networks is given in Chapter 12 of [11] (a solution for the special case of a fluid reentrant line is given in [37]). For completeness, we state the result here. We start by calculating the total workload in the buffers.

The amount that should be emptied from the buffers until all buffers are drained, can be written in two parts. The first part is the amount of fluid that flows from the buffers due to the initial fluid inventory and is given by:

$$a + P'a + (P')^2 a + (P')^3 a + \cdots = Qa.$$

The second part is the amount of fluid which arrives to the system exogenously up to time t, and is given by:

$$\alpha t + P'\alpha t + (P')^2 \alpha t + \cdots = Q\alpha t, \quad \forall t \in [0, \infty).$$

Using the calculations above, we can compute the total cumulative workload for buffer k up to time t as

$$(Q^k a + Q^k \alpha t) m_k. \tag{2.4}$$

We are now prepared to present the solution to the deterministic makespan problem.

Theorem 2.2.1. *If the usual traffic conditions for a multiclass fluid network hold then a lower bound for the makespan is:*

$$t^* \geq t^{LB} = \max_{j \in J} \left\{ \frac{\sum_{k \in \mathfrak{C}_j} Q^k a m_k}{1 - \sum_{k \in \mathfrak{C}_j} Q^k \alpha m_k} \right\}, \tag{2.5}$$

and this value can be attained. Conversely, if the usual traffic conditions are violated then the makespan is infinite for every policy.

Proof. See Chen and Yao [11], page 384. □

If a numerator in (2.5) is zero (i.e., there is no initial fluid to be processed at j) then the associated ratio is zero, regardless of the value of the denominator. Otherwise, if the usual traffic conditions hold but they do not hold strictly then the lower bound (2.5) is infinite. In this case, we regard the lower bound as being attained, as stated in Theorem 2.2.1, since the makespan is also infinite.

As Theorem 2.2.1 shows, the deterministic makespan problem is analytically solvable. However, to deal with more general cases, i.e., the cases where parameters are random, we need to formulate the makespan problem as a mathematical programming model. We have already introduced one set of structural constraints for our problem, namely (2.2). The next step in formulating the problem is to mathematically represent the makespan in terms of v and the parameters (a, α, m). To do so, we use the fact that if the total workload of buffer k is to be drained at time t_k, the associated effort expended must equal the amount of work that has arrived to buffer k up to time t_k, that is,

$$v_k t_k = Q^k a m_k + Q^k \alpha m_k t_k, \quad \forall k \in K.$$

Solving this equation for t_k we obtain,

$$t_k = \frac{Q^k a m_k}{v_k - Q^k \alpha m_k}, \quad \forall k \in K. \tag{2.6}$$

To interpret the expression given in (2.6), observe that the denominator gives the remaining percentage of effort available, if all the work for buffer k

due to exogenous arrivals is removed from the system as soon as it arrives. The t_k given in (2.6) is the time at which buffer k has processed all of the work initially present in the system, assuming it drains the workload due to exogenous arrivals as soon as it arrives. Therefore the makespan of the system under allocation v is the time when all buffers empty their initial amounts from the system, and is given by:

$$MS(v, a, \alpha, m) = \max_{k \in K} \left\{ \frac{Q^k a m_k}{[v_k - Q^k \alpha m_k]^+} \right\}. \tag{2.7}$$

Here, the fact that the policy is a stationary policy is key. We see that under policy v, $\frac{Q^k a m_k}{[v_k - Q^k \alpha m_k]^+}$ is the minimal time that buffer k can drain its initial workload in the system, so taking the maximum over all buffers, we obtain a lower bound for the makespan. Another observation is that, since the allocations are constant, each buffer is fed by a constant flow and when the buffer is empty it stays empty forever. And, in the worst case, where the total workload is initially present in the buffer, buffer k would drain this work by time $\frac{Q^k a m_k}{[v_k - Q^k \alpha m_k]^+}$. Hence, taking the maximum over all buffers, we obtain an upper bound on the makespan under v. This shows that equation (2.7) characterizes the makespan of the system.

A necessary condition to have a finite expected makespan under a stationary policy is

$$v_k \geq Q^k \alpha m_k, \quad \forall k \in K. \tag{2.8}$$

The inclusive inequality in constraint (2.8) allows for the possibility of infinite makespan, but we minimize with respect to decision vector v and so the expected makespan will be finite whenever possible in the optimization problem. Also, note that nonnegativity of the allocation decisions, $v \geq 0$, is ensured by (2.8).

Summarizing the development in (2.2), (2.7) and (2.8), we obtain the following formulation for the deterministic makespan problem:

$$\begin{aligned} \min_{v} \quad & \max_{k \in K} \left\{ \frac{Q^k a m_k}{v_k - Q^k \alpha m_k} \right\} && \text{(2.9a)} \\ \text{s.t.} \quad & \sum_{k \in \mathcal{C}_j} v_k \leq 1, \quad \forall j \in J && \text{(2.9b)} \\ & v_k \geq Q^k \alpha m_k, \ \ \forall k \in K. && \text{(2.9c)} \end{aligned}$$

We now show that the deterministic makespan problem is *separable* by station.

Theorem 2.2.2. *If the strict usual traffic conditions hold, then the deterministic makespan problem (2.9) can be solved by solving the station makespan problems separately. Specifically, let*

$$\bar{v}^j \in \arg\min_{V^j} \max_{k \in \mathcal{C}_j} \left\{ \frac{Q^k a m_k}{v_k - Q^k \alpha m_k} \right\}, \ \ j \in J, \qquad (2.10)$$

where $V^j = \{[v_k]_{k \in \mathcal{C}_j} : \sum_{k \in \mathcal{C}_j} v_k \leq 1, v_k \geq Q^k \alpha m_k, k \in \mathcal{C}_j\}$. *Then,* $v^* = [\bar{v}^j]_{j \in J}$ *solves (2.9).*

Proof. Constraints (2.9b) and (2.9c) are equivalent to $[v^j]_{j \in J} \in \prod_{j \in J} V^j$. The usual traffic conditions (2.3) ensure $V^j \neq \emptyset, \forall j \in J$, and hence that (2.9) is feasible. Strictness of these conditions ensures a finite makespan. Formulation

(2.9) can be written as:

$$\begin{aligned}\min_{[v^j \in V^j]_{j\in J}} \max_{k\in K} \left\{ \frac{Q^k a m_k}{v_k - Q^k \alpha m_k} \right\} &= \min_{[v^j \in V^j]_{j\in J}} \max_{j\in J} \max_{k\in \mathcal{C}_j} \left\{ \frac{Q^k a m_k}{v_k - Q^k \alpha m_k} \right\} \\ &= \max_{j\in J} \min_{v^j \in V^j} \max_{k\in \mathcal{C}_j} \left\{ \frac{Q^k a m_k}{v_k - Q^k \alpha m_k} \right\}. \quad (2.11)\end{aligned}$$

The inner minimization in (2.11) is equivalent to that in (2.10), and the proof is complete. $\square$

Theorem 2.2.2 shows that even if the output of one station provides input to another station, the optimal allocations of effort can be determined separately. In the next chapter, we show that Theorem 2.2.2 does not extend in general to the stochastic makespan problem even when there is no fluid flow between stations.

Chapter 3

Stochastic Makespan Problem Without Recourse Action

This chapter is devoted to the stochastic makespan problem under the first decision scheme given in the introduction. Under this decision scheme, the controller has to decide on the allocations v_k's without prior knowledge of the parameters. We assume that, even though the parameters a, α and m are not known at the time of decision making, the controller possesses distributional information about the parameters. After the allocations are set at time 0, the controller cannot change them no matter the realization of the parameters. In other words, the controller is not allowed to take a recourse action after the parameters of the system are revealed. This decision making structure is especially reasonable, when the allocations are results of a structural decision. The controller's goal is to minimize the expected value of the makespan.

In the case where parameters are known deterministically, we can find a closed-form expression for an optimal policy. It can easily be shown that one optimal policy in the deterministic problem is a stationary policy. However, in this work our main focus is on the makespan problem where the parameters a, α and m are only known via a probability distribution, and in this case we restrict the control *a priori* to stationary policies.

The mathematical formulation for this decision structure is a simple modification of the model given in (2.9). After the parameters in (2.9) are replaced with random variables, the objective function becomes a random function. Hence, the objective function should be replaced with the expectation of this random function, i.e.,

$$\begin{aligned}
\min_{v} \quad & \mathbb{E}\left(\max_{k\in K}\left\{\frac{Q^k\tilde{\alpha}\tilde{m}_k}{v_k - Q^k\tilde{\alpha}\tilde{m}_k}\right\}\right) && \text{(3.1a)}\\
\text{s.t.} \quad & \sum_{k\in\mathcal{C}_j} v_k \le 1, \qquad \forall j \in J && \text{(3.1b)}\\
& v_k \ge Q^k\tilde{\alpha}\tilde{m}_k, \ \ \forall k \in K, \ \text{w.p.1.} && \text{(3.1c)}
\end{aligned}$$

Constraints (3.1c) slightly differs from (2.9c). The $[\cdot]^+$ function given in the denominator of (2.7) is omitted in (3.1a). Hence, unless the constraints (3.1c) are imposed, the objective function does not represent the true makespan of the system. Physically this means that to be able to obtain a finite expected makespan, we should at least be able to drain the incoming workload with probability 1.

We observed some desirable properties of the deterministic makespan problem in Chapter 2. Hence, the following questions naturally arise with regard to the stochastic makespan problem:

(i) Is it possible to solve the stochastic makespan problem by solving a deterministic problem using the expected values of the parameters? If not, what can be said about the suboptimality of this deterministic solution?

(ii) Theorem 2.2.2 shows that the deterministic problem can be solved for stations separately. Does this property also hold for the stochastic problem?

(iii) The deterministic problem is well-posed, i.e., there exists a solution with finite makespan, when the usual traffic conditions are satisfied. What are the well-posedness conditions for the stochastic problem?

(iv) What are the solution methodologies that can be used to solve the stochastic makespan problem?

In Section 3.1, we address (i) and (ii). By counterexamples, we show that, solving a deterministic version of the problem may lead to drastically suboptimal solutions when the parameters are stochastic. We also demonstrate that the separability property does not hold for the stochastic problem, even when the stations are completely "independent." In Section 3.2, we address (iii) and develop the well-posedness conditions for the stochastic makespan problem. Finally, in Section 3.3, we investigate the stochastic programming methodologies that can be used to solve the stochastic problem.

3.1 Counterexamples

Perhaps the simplest approach to the stochastic makespan problem is to attempt to solve the problem as in the deterministic case, using the expected values of the stochastic parameters, i.e., we use the solution for the deterministic makespan problem, where a, α and m are replaced by their population means. This solution is called the expected value solution. Despite being commonly used in practice, the expected value solution is in general suboptimal when one of the parameter vectors a, α or m is random.

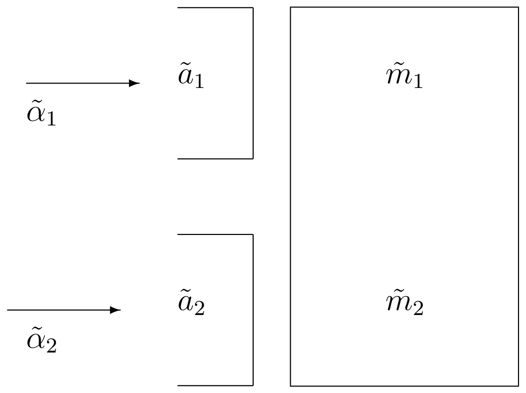

Figure 3.1: Network with One Station and Two Buffers

Consider the network with one station and two buffers shown in Figure 3.1. Assume that the parameters a and m are known deterministically at the time of decision making and $a = (3, 2)$ and $m = (5/24, 5/24)$. The incoming rates are random with $\mathbb{P}(\tilde{\alpha} = (3/2, 3/2)) = \mathbb{P}(\tilde{\alpha} = (1/2, 5/2)) = 1/2$. The expected value solution $v_{EV} = (0.433, 0.567)$ results in an expected makespan of 7.13. However, if $v = (0.404, 0.596)$, the expected makespan is 6.19, which shows the expected value solution, v_{EV}, is suboptimal.

Next suppose m is random in the network of Figure 3.1 with $\mathbb{P}(\tilde{m} = (4/24, 4/24)) = \mathbb{P}(\tilde{m} = (6/24, 6/24)) = 1/2$ and $a = (3, 2)$ and $\alpha = (1, 2)$. The optimal solution of the expected value problem, v_{EV}, has an expected makespan of 4.69. However, for $v = (0.4, 0.6)$, the expected makespan is 3.57.

Finally, suppose a is random with $\mathbb{P}(\tilde{a} = (1,1)) = \mathbb{P}(\tilde{a} = (5,3)) = 1/2$ and $\alpha = (1,2)$ and $m = (5/24, 5/24)$. In this case, using v_{EV} we obtain an expected makespan of 3.01. When $v = (0.443, 0.557)$, the expected makespan is 2.95, which is again lower than that of the expected value solution.

The examples above illustrate that the expected value solution is suboptimal in general. At this point, it is natural to ask whether there exist any bounds on the suboptimality of the expected value solution. Unfortunately, the expected value solution can be drastically suboptimal when one of the parameter vectors a, α or m is random. We now show that the expected value solution may lead to an infinite expected makespan, even if there are feasible solutions where the expected makespan is finite.

Consider the network in Figure 3.1. Let $a = (0,6)$, $m = (1,1)$ and α be random with $\mathbb{P}(\tilde{\alpha} = (0,0)) = \mathbb{P}(\tilde{\alpha} = (1/4,1/4)) = 1/2$. As a result, $v_{EV} = (1/8, 7/8)$ is the expected value solution, i.e., the effort allocated for buffer 1 is just enough to serve the expected inflow and the rest is devoted to drain buffer 2. It is easy to see that with probability 1/2, this solution does not drain the system (i.e., when the scenario $\tilde{\alpha} = (1/4, 1/4)$ occurs). Hence, the expected makespan is infinite if v_{EV} is employed. However, the solution $v^* = (1/4, 3/4)$ yields an expected makespan of 10. It is possible to find a similar example when only m is random. For the case in which a is random, the expected value solution yields a finite makespan with probability 1, if there is a feasible solution with finite expected makespan.

When parameters of the system are deterministically known, Theorem 2.2.2 shows that even if the output of one station provides input to another station, the optimal allocations of effort can be determined separately. Unfortunately, Theorem 2.2.2 does not extend in general to the stochastic makespan problem. For stochastic problems, the following example shows that this separation result fails to hold even when there is no fluid flow between the stations.

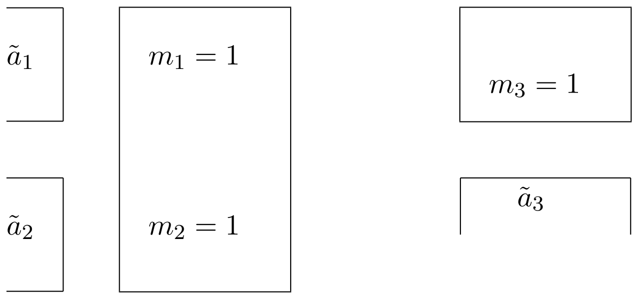

Figure 3.2: Non-Separable Stochastic Network

Consider the two-station three-buffer network in Figure 3.2, where there is no input, only the initial inventory is random, and there is no flow between buffers. The service times are as given in the figure, and let $\mathbb{P}(\tilde{a} = (5, 1, 100)) = \mathbb{P}(\tilde{a} = (1, 5, 0)) = 1/2$. Obviously, for the second station we allocate $v_3 = 1$. If we solve the stochastic problem for station one without considering the second station, we obtain $v_1 = v_2 = 1/2$, which leads to an expected network makespan of 55. However, setting $v = (1/6, 5/6, 1)$, yields an expected makespan of 53.

In the example above, the suboptimality of the allocation based on optimizing the stations separately arises as a result of dependency in the random vector governing the initial inventory. Next, suppose that $\mathbb{P}(\tilde{a} = (10, 0, 14)) =$

$\mathbb{P}(\tilde{a} = (0,4,14)) = 1/2$. In this problem, a_3 is deterministic and therefore independent of (a_1, a_2). When the problem is solved for the stations separately we obtain the optimal allocation as $v^* = (0.613, 0.387, 1)$, which yields an expected makespan of 15.15. However, when the problem is solved taking both stations into account the optimal allocation is $v^* = (0.714, 0.286, 1)$, yielding a makespan of 14. Therefore the separability property does not necessarily hold, even when the stations are completely independent.

Equation (2.11) in the proof of Theorem 2.2.2 follows from exchanging the order of minimization and maximization. However, when we have stochastic parameters, the analog of equation (2.11) is:

$$\min_{[v^j \in V^j]_{j\in J}} \mathbb{E}\left(\max_{k\in K} \left\{ \frac{Q^k a m_k}{v_k - Q^k \alpha m_k} \right\} \right) = \min_{[v^j \in V^j]_{j\in J}} \mathbb{E}\left(\max_{j\in J} \max_{k\in \mathcal{C}_j} \left\{ \frac{Q^k a m_k}{v_k - Q^k \alpha m_k} \right\} \right).$$

With the presence of the expectation operator, it is no longer possible to interchange the optimization operations. As a result, Theorem 2.2.2 does not hold for the stochastic case.

3.2 Existence of a Finite Optimal Solution

For the deterministic makespan problem, Theorem 2.2.1 implies that, if the usual traffic conditions are strictly satisfied, a solution that yields a finite makespan exists. The natural question that then arises in the stochastic problem is whether there is a solution which yields a finite expected makespan if the usual traffic conditions hold with probability 1. Unfortunately, it turns out that the almost sure usual traffic conditions do not, in general, guarantee a finite expected makespan in the stochastic makespan problem. As an example consider the network in Figure 3.1 when α is random. Suppose a and m are deterministic with $a = (0, 0)$ and $m = (1, 1)$, and let $\mathbb{P}(\tilde{\alpha} = (2/3, 0)) = \mathbb{P}(\tilde{\alpha} = (0, 2/3)) = 1/2$. The traffic intensity, $\sum_k \tilde{\alpha}_k m_k$, is 2/3 under both scenarios. So, the usual traffic conditions are satisfied in both scenarios. However, since $\tilde{\alpha}_1$ takes value 2/3 in the first scenario, it is necessary to have $v_1 \geq 2/3$ for a finite expected makespan. However, by a symmetric argument we also need $v_2 \geq 2/3$ for the second scenario. These conditions on (v_1, v_2) are inconsistent with (3.1b), i.e., there is no effort allocation which yields a finite expected makespan, even though the usual traffic conditions are satisfied for each of the scenarios.

Motivated by the above example, we now derive necessary and sufficient conditions which guarantee the existence of an allocation with finite expected makespan. To do so, we need the notion of the *essential supremum* of a random variable $\tilde{X}$:

$$\operatorname{ess\,sup}\{\tilde{X}\} \equiv \inf\{x : \mathbb{P}(\tilde{X} > x) = 0\}.$$

Theorem 3.2.1. *Consider a fluid makespan problem with stochastic parameters, and let $S_k \equiv \operatorname{ess\,sup}\{Q^k \tilde{\alpha} \tilde{m}_k\}$. The necessary and sufficient conditions for a solution with a finite expected makespan to exist are:*

(a) $\sum_{k \in \mathcal{C}_j} S_k \leq 1$, $\forall j \in J$

(b) $\mathbb{E}\left(\dfrac{Q^k\tilde{a}\tilde{m}_k}{S_k + \dfrac{1-\sum_{l\in\mathcal{C}_{\sigma_k}} S_l}{|\mathcal{C}_{\sigma_k}|} - Q^k\tilde{\alpha}\tilde{m}_k}\right) < \infty$, $\forall k \in K$.

Proof. If conditions (a) and (b) hold then allocation vector v, where $v_k = S_k + \dfrac{1-\sum_{l\in\mathcal{C}_{\sigma_k}} S_l}{|\mathcal{C}_{\sigma_k}|}$ for all $k \in K$, satisfies (3.1b) and (3.1c). From (b) we know that

$$\mathbb{E}\left(\frac{Q^k\tilde{a}\tilde{m}_k}{v_k - Q^k\tilde{\alpha}\tilde{m}_k}\right) < \infty, \quad \forall k \in K.$$

The buffer index set K is finite and hence so is the objective function (3.1a) evaluated at this v. This proves the sufficiency of conditions (a) and (b) for the existence of a solution with a finite expected makespan.

Now, suppose that (a) does not hold, i.e., there is a station $j^* \in J$ with $\sum_{k\in\mathcal{C}_{j^*}} S_k > 1$. Constraint (3.1c) is equivalent to $v_k \geq S_k$, for all $k \in K$. Thus, (3.1b) for $j = j^*$ and (3.1c) for $k \in \mathcal{C}_{j^*}$ are inconsistent, i.e., (3.1) is infeasible and hence has no solution with finite expected makespan.

Finally, suppose that (b) does not hold, i.e.,

$$\mathbb{E}\left(\frac{Q^{k^*}\tilde{a}\tilde{m}_{k^*}}{S_{k^*} + \dfrac{1-\sum_{l\in\mathcal{C}_{\sigma_{k^*}}} S_l}{|\mathcal{C}_{\sigma_{k^*}}|} - Q^{k^*}\tilde{\alpha}\tilde{m}_{k^*}}\right) = \infty \tag{3.2}$$

for some $k^* \in K$. If (3.1b)-(3.1c) is infeasible then the proof is complete so we restrict attention to the case when $\sum_{l\in\mathcal{C}_{\sigma_{k^*}}} S_l \leq 1$, and, in turn, consider the cases when this inequality holds with equality and is strict.

Case 1: $\sum_{l \in \mathcal{C}_{\sigma_{k^*}}} S_l = 1$. In this case, in all feasible allocations, $v_{k^*} = S_{k^*}$. By (3.2),

$$\mathbb{E}\left(\frac{Q^{k^*}\tilde{a}\tilde{m}_{k^*}}{S_{k^*} - Q^{k^*}\tilde{\alpha}\tilde{m}_{k^*}}\right) = \infty,$$

and hence the objective function (3.1a) is also infinite for any feasible v.

Case 2: $\sum_{l \in \mathcal{C}_{\sigma_{k^*}}} S_l < 1$. So, there exists an $\epsilon > 0$ such that $\sum_{l \in \mathcal{C}_{\sigma_{k^*}}} S_l + \epsilon = 1$. Thus,

$$\mathbb{E}\left(\frac{Q^{k^*}\tilde{a}\tilde{m}_{k^*}}{S_{k^*} + \frac{\epsilon}{|\mathcal{C}_{\sigma_{k^*}}|} - Q^{k^*}\tilde{\alpha}\tilde{m}_{k^*}}\right) \leq \frac{|\mathcal{C}_{\sigma_{k^*}}|\mathbb{E}\left(Q^{k^*}\tilde{a}\tilde{m}_{k^*}\right)}{\epsilon},$$

which implies $\mathbb{E}\left(Q^{k^*}\tilde{a}\tilde{m}_{k^*}\right) = \infty$. For any feasible allocation v,

$$\mathbb{E}\left(\frac{Q^{k^*}\tilde{a}\tilde{m}_{k^*}}{v_{k^*} - Q^{k^*}\tilde{\alpha}\tilde{m}_{k^*}}\right) \geq \mathbb{E}\left(\frac{Q^{k^*}\tilde{a}\tilde{m}_{k^*}}{1}\right) = \infty.$$

□

When the conditions (a) and (b) of Theorem 3.2.1 are assumed to hold, the proof of the theorem also characterizes a feasible solution which yields a finite expected makespan. We assume the existence of a finite optimal solution for the remainder of this work.

3.3 Solution Methods

In Section 3.1, we show that the stochastic makespan problem can not be solved using the expected values of the parameters. So, in this section we outline methods for solving, or approximately solving, the stochastic makespan problem (3.1), which are more generally applicable. Our goal here is to give an overview of available solution approaches, depending on the nature of the distribution of $\tilde{\xi} = (\tilde{a}, \tilde{\alpha}, \tilde{m})$, not to carry out a detailed computational study. However, we do provide results that suggest our optimization model (3.1) is numerically tractable on moderate- to large-sized networks. (Our test problems are reentrant lines with up to 75 buffers and 15 stations.)

First we note that for fixed ξ, $h_k(v_k, \xi) = \frac{Q^k a m_k}{v_k - Q^k \alpha m_k}$ is convex in feasible v_k, and hence, so are $MS(v, \xi) = \max_{k \in K} h_k(v_k, \xi)$ and $\mathbb{E}\left(MS(v, \tilde{\xi})\right)$. This, coupled with the fact that (3.1c) can be replaced by $v_k \geq S_k \equiv \operatorname{ess\,sup}\{Q^k \tilde{\alpha} \tilde{m}_k\}$, $\forall k \in K$, means that (3.1) is a convex optimization problem. If Ω is finite with a modest number of sample points and with probability mass function $p^\omega = \mathbb{P}(\tilde{\xi} = \xi(\omega))$, $\omega \in \Omega$, then we can solve (3.1) using a convex nonlinear programming algorithm. One such algorithm is a variant of Kelley's cutting-plane method [26] that handles the nondifferentiability of our objective function that arises from the "$\max_{k \in K}$." In stochastic programming, this algorithm is known as the L-shaped method [36]. The algorithm iteratively solves a master program whose size depends on the dimension of v and evaluates $\mathbb{E}\left(MS(v, \tilde{\xi})\right)$ and its (sub)gradient at the master program solution. The algorithm scales well with $|\Omega|$ because these latter computations separate for each $\omega \in \Omega$, and hence, can be done quickly.

If $\tilde{\xi}$ has too many (possibly an infinite number of) realizations, we cannot solve (3.1) exactly but approximation techniques can be employed. We discuss two approximations: one based on Monte Carlo sampling and the other on deterministically-valid bounds.

The Monte Carlo sampling approximation, in its simplest form, entails generating independent and identically distributed (i.i.d.) observations $\tilde{\xi}^1, \ldots, \tilde{\xi}^n$ from the distribution of $\tilde{\xi}$ and solving

$$\begin{aligned}
\min_{v} \quad & \frac{1}{n}\sum_{i=1}^{n} \max_{k\in K} h_k(v_k, \tilde{\xi}^i) && && \text{(3.3a)}\\
\text{s.t.} \quad & \sum_{k\in \mathcal{C}_j} v_k \le 1, && \forall j \in J && \text{(3.3b)}\\
& v_k \ge S_k && \forall k \in K. && \text{(3.3c)}
\end{aligned}$$

Let $v^*(n)$ denote the optimal solution and $z^*(n)$ denote the optimal objective function value of (3.3). Model (3.1) has a convex objective function and a compact (and convex) feasible region. As a result, with probability one: $z^*(n) \to z^*$, where z^* is the optimal value of (3.1), and all limit points of $\{v^*(n)\}$ solve (3.1). See the recent review in [34] for these consistency results and other asymptotic properties of the Monte Carlo method. Of course, from the perspective of numerically solving model (3.3), we may again exploit its structure with application of the cutting-plane method described above.

Our second approximation method applies to the special cases of the stochastic makespan problem, when only one set of the stochastic parameters (either a, α or m) is random or when these three random vectors are independent. This approach uses deterministically-valid bounds on the objective function which exploit the convexity of $MS(v, \cdot)$ with respect to the stochastic parameters. Hence, to employ the bounds, we shall first prove that the objective function is convex with respect to the stochastic parameters.

Theorem 3.3.1. *Let v be a feasible allocation satisfying (3.3b) and (3.3c). Then $MS(v,\cdot,\alpha,m)$, $MS(v,a,\cdot,m)$ and $MS(v,a,\alpha,\cdot)$ are convex functions on the convex hull of the stochastic parameters' support.*

Proof. It suffices to show $h_k(v_k,\cdot,\alpha,m)$, $h_k(v_k,a,\cdot,m)$ and $h_k(v_k,a,\alpha,\cdot)$ are convex because in each case, MS is then the maximum of a finite collection of convex functions, and hence is convex.

Case 1: $h_k(v_k,\cdot,\alpha,m)$ is a linear function and thus convex.

Case 2: $h_k(v_k,a,\cdot,m)$ is the composition of a convex, increasing function, $f(x)=\frac{Q^k a m_k}{v_k-x}$, with a linear function, and is therefore convex.

Case 3: Let $f(m_k)=\frac{Q^k a m_k}{v_k-Q^k\alpha m_k}$. The second derivative of f is:

$$\frac{d^2 f(m_k)}{dm_k^2}=\frac{Q^k a Q^k \alpha v_k}{(v_k-Q^k\alpha m_k)^3}.$$

Convexity of f, and hence $h_k(v_k,a,\alpha,\cdot)$, again follows as v_k is feasible. □

Let $f:\Re^d\to\Re$ be a convex function and $\tilde{\xi}$ be a random d-vector. Jensen's inequality provides a well-known lower bound on $\mathbb{E}f(\tilde{\xi})$, i.e., $\mathbb{E}f(\tilde{\xi})\geq f(\mathbb{E}\tilde{\xi})$. When $\tilde{\xi}$ has bounded support, a class of upper bounds on $\mathbb{E}f(\tilde{\xi})$ is provided by the Edmundson-Madansky (EM) inequality. Madansky [29] and Frauendorfer [16] develop this bound in the respective cases when the components of $\tilde{\xi}$ are independent and dependent, assuming that $\tilde{\xi}$'s support is (contained in) a hyper-rectangle. These results have been extended to simplicial and general polyhedral domains [14, 18]. We represent an EM bound via $\mathbb{E}f(\tilde{\xi}) \leq \mathbb{E}f(\tilde{\xi}_{EM})$, where $\tilde{\xi}_{EM}$ is a random vector taking values only at the extreme points of $\tilde{\xi}$'s support. So, if the domain is a hyper-rectangle, computing $\mathbb{E}f(\tilde{\xi}_{EM})$ requires 2^d evaluations of f but that number is $d+1$ for a simplicial domain.

Theorem 3.3.1 allows us to apply the bounds of Jensen and Edmundson-Madansky to the following important special cases of the stochastic makespan problem.

Corollary 3.3.1. Let $MS(v, \xi)$ denote the makespan function. If only one set of the stochastic parameters (either a, α or m) is random or if the subvectors $\tilde{a}$, $\tilde{\alpha}$ and $\tilde{m}$ of $\tilde{\xi} = (\tilde{a}, \tilde{\alpha}, \tilde{m})$ are mutually independent then

$$MS(v, \mathbb{E}\tilde{\xi}) \leq \mathbb{E}\left(MS(v, \tilde{\xi})\right) \leq \mathbb{E}\left(MS(v, \tilde{\xi}_{EM})\right). \tag{3.4}$$

Assuming $\tilde{\xi}$ satisfies the hypothesis of Corollary 3.3.1, we can solve the makespan problem under the single scenario $\mathbb{E}\tilde{\xi}$ to obtain allocation v_{EV} and optimal value $\underline{z} = MS(v_{EV}, \mathbb{E}\tilde{\xi}) \leq z^* \equiv \min_v \mathbb{E}\left(MS(v, \tilde{\xi})\right)$. Carrying out this optimization over the feasible region of (3.3b)-(3.3c) ensures v_{EV} is feasible to the stochastic makespan problem. Then, we compute $\bar{z} = \mathbb{E}\left(MS(v_{EV}, \tilde{\xi}_{EM})\right) \geq \mathbb{E}\left(MS(v_{EV}, \tilde{\xi})\right) \geq z^*$. If $\bar{z} - \underline{z}$ is sufficiently small then v_{EV} is a high quality approximate solution to the stochastic makespan problem. Otherwise, the Jensen and Edmundson-Madansky bounds can be tightened by applying them in conditional fashion to a partition of $\tilde{\xi}$'s support. In this way, the lower and upper bounds of (3.4) allow us to employ a bounding-and-approximation scheme to (approximately) solve the stochastic makespan problem.

We apply our two approximation schemes to four test problems, which are reentrant lines with buffer-station combinations of 10-5, 25-5, 50-10 and 75-15. In the "10-5" test problem, fluid makes two left-to-right passes through the 5 stations, while in the other three problems the fluid makes 5 such passes. Parameters a and m are deterministic. The incoming rates are zero except at the first buffer of the first station, and $\tilde{\alpha}_1$ is assumed to be a continuous uniform random variable on $(0, \alpha_1^{\max})$. In the 75-15 model, for example, we form a random test problem by selecting 75 m_k values uniformly from $[0, 1]$ and we similarly select 75 a_k values from the discrete uniform on $\{1, \ldots, 10\}$. The value of $\alpha_1^{\max}$ is then selected so that the usual traffic conditions hold strictly.

We apply the Jensen and Edmundson-Madansky bounds conditionally to a partition of $(0, \alpha^{\max})$ with $n=$10,000 equally-sized cells. We compute the associated Jensen bound by solving the stochastic makespan problem with 10,000 realizations, i.e., conditional expectations on the 10,000 cells, using the cutting-plane method described above. Doing so, yields $\underline{z}$ and a solution $v^*(n)$. We then evaluate the Edmundson-Madansky upper bound, $\bar{z}$, at $v^*(n)$, which requires $10,001$ function evaluations of $MS(v^*(n), \cdot)$. For the four test problems, the associated percentage gap, $100 \cdot (\bar{z} - \underline{z})/\underline{z}$ is listed in Table 3.1. By

Buffers-Stations	EM-J
10-5	0.0000
25-5	0.0420
50-10	0.0004
75-15	0.0046

Table 3.1: Gap between the Edmundson-Madansky and Jensen bounds for reentrant lines of different sizes for 10,000 partitions

increasing the number of cells n we can ensure that the conditional Jensen and Edmundson-Madansky bounds, as well as the solutions $v^*(n)$, converge to their counterparts for problem (3.1). Development of this sequential approximation method using the Jensen and EM bounds begins with Huang, Ziemba and Ben-Tal [22], and adaptive schemes for forming the cell-based partition of the support are described, e.g., by [10, 15, 17, 25].

Table 3.2 shows the computation time required to solve instances of model (3.3) for n=10,000 i.i.d. observations of $\tilde{\alpha}$ to varying levels of precision, again using the cutting-plane algorithm. The reported CPU times (in seconds) are on a 1.8 GHz, Pentium Xeon dual-processor machine with 1 GB of memory. At each iteration the algorithm produces upper and lower bounds $\bar{z}^*(n)$ and $\underline{z}^*(n)$ on $z^*(n)$, the optimal value of model (3.3). The cutting-plane algorithm terminates when $(\bar{z}^*(n) - \underline{z}^*(n))/\underline{z}^*(n) \leq \epsilon$. The coefficient of variation of the sample mean objective function of the 75-15 test problem, with $v = v^*(n)$ and n=10,000, is roughly 10^{-5} meaning that there is little point in solving model (3.3) for more precise values of ϵ.

We note that the computation times for the Jensen lower-bound with

Table 3.2: Computation times (seconds) for reentrant lines of different sizes for 10,000 sample points

Buf-Stat	$\epsilon = 10^{-1}$	$\epsilon = 10^{-2}$	$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$
10-5	2	2	3	4	4	5
25-5	22	30	60	213	411	518
50-10	89	141	334	1469	4263	5331
75-15	208	861	1877	3210	6967	11212

10,000 cells are essentially the same as those reported in Table 3.2. These computations indicate that it is possible to solve the stochastic makespan problem for large networks with a desirable level of accuracy in reasonable time.

Chapter 4

Stochastic Makespan Problem with Recourse Action

In this chapter, we consider the stochastic makespan problem when the decision-maker can take a recourse action. The decision-maker has to decide on the percentage of effort that will be allocated to each buffer served at each station without prior knowledge of the system parameters. It is assumed that the decision-maker only has distributional knowledge at this initial phase. After these allocations are set, the system starts running. After the system has operated for sometime, $\tilde{T}$, the randomness is revealed, i.e., the decision-maker learns the parameters and changes the allocations based on this knowledge. The time at which the randomness is revealed ($\tilde{T}$) can be random, and models the lead time it takes us to adapt the system design, i.e., modify the allocation of effort at each station. The decision-making structure is shown in Figure 4.1.

For a particular realization of the random parameters and allocation vector, there are two possible cases. In the first case, the system is drained by time $\tilde{T}$, and hence, no recourse action is needed. In the second case, there is fluid present in some of the buffers at time $\tilde{T}$, and the decision-maker takes a recourse action.

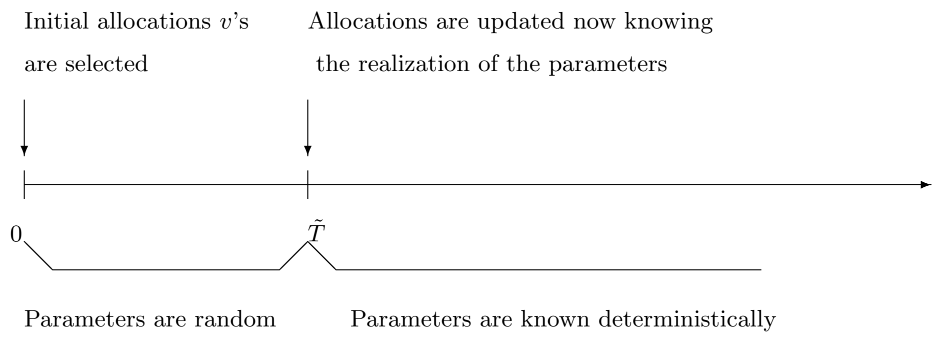
Initial allocations v's are selected
Allocations are updated now knowing the realization of the parameters
0
$\tilde{T}$
Parameters are random
Parameters are known deterministically

Figure 4.1: The timeline for making decisions

For the first case, for a given realization ω and a given initial allocation vector v, the system is drained by time $T(\omega)$ and the decision-maker does not have any chance to take a recourse action. Hence, for $\xi(\omega) = (a(\omega), \alpha(\omega), m(\omega), T(\omega))$ and v, we treat the problem as in the no-recourse case and calculate the makespan using the function

$$f_1(v, \xi(\omega)) = \max_{k \in K} \left\{ \frac{Q^k a(\omega) m_k(\omega)}{[v_k - Q^k \alpha(\omega) m_k(\omega)]^+} \right\}. \tag{4.1}$$

The characterization of the makespan in the case that system is not drained by time $\tilde{T}$ is somewhat more complicated. After the parameters are revealed at time $\tilde{T}$, the decision-maker can utilize Theorem 2.2.1 to minimize the makespan for the remaining workload. Hence, the determination of the remaining workload at time $\tilde{T}$ plays a major role in the optimization process. Given a particular realization ω and initial allocations v, (4.2) characterizes the workload at time $T(\omega)$ and calculates the makespan in the case the system drains after time $T(\omega)$. Dependence on ω is suppressed to simplify the notation:

$$
\begin{aligned}
f_2(v,\xi) \quad &= \min_{x} \quad \max_{j \in J} \left\{ \frac{\sum_{k \in \mathcal{C}_j} Q^k (a + \alpha T) m_k - x_k m_k}{1 - \sum_{k \in \mathcal{C}_j} Q^k \alpha m_k} \right\} + T && \text{(4.2a)} \\
& \qquad\qquad m_k x_k \le v_k T, && \forall k \in K \quad \text{(4.2b)} \\
& \qquad\qquad x_k \le a_k + \alpha_k T + \sum_{l \in K} p_{lk} x_l, && \forall k \in K. \quad \text{(4.2c)}
\end{aligned}
$$

In model (4.2), x_k is the actual amount of fluid drained from buffer k by time $T(\omega)$. The first term in the objective function is the optimal draining time after time $T(\omega)$, obtained by applying Theorem 2.2.1 to the remaining inventory. To find the overall draining time, we add $T(\omega)$. The constraints (4.2b) ensures that the actual rate the workload is drained from buffer k does not exceed the allocated capacity. If fluid is not present in buffer k, then the allocated capacity for that buffer will not be fully utilized, and constraints (4.2c) handle this issue. The right-hand sides of these constraints are the amount of workload that buffer k can work on from time 0 to $T(\omega)$. Notice that the elements of the $|K|$-dimensional vector $(a(\omega) + \alpha(\omega)T(\omega) + (I - P')xT(\omega))$ are the amount of fluid in each buffer at time $T(\omega)$. Multiplying appropriately by Q^k and $m_k(\omega)$, and doing the necessary cancelations we obtain (4.2a).

The objective function in (4.2a) is a piecewise linear convex function in x. We use a standard trick to convert (4.2) to a linear program. By replacing the objective function with an auxiliary variable θ, we obtain:

$$
\begin{aligned}
f_2(v,\xi) \quad &= \min_{\theta,x} \quad \theta && \text{(4.3a)}\\
& \qquad m_k x_k \le v_k T, && \forall k \in K \text{(4.3b)}\\
& \qquad x_k \le a_k + \alpha_k T + \sum_{l\in K} p_{lk} x_l, && \forall k \in K \text{ (4.3c)}\\
& \qquad \theta \ge \frac{\sum_{k\in\mathcal{C}_j} Q^k(a+\alpha T)m_k - x_k m_k}{1-\sum_{k\in\mathcal{C}_j} Q^k \alpha m_k} + T, && \forall j \in J. \text{ (4.3d)}
\end{aligned}
$$

As a result of (4.3c) the first term on the right hand side of (4.3d) cannot be less than 0. We can conclude that $f_2(v,\xi(\omega)) \ge T(\omega)$ for all v and ω. Hence, when the system is drained before time $T(\omega)$, $f_1(v,\xi(\omega)) \le f_2(v,\xi(\omega))$. Using Theorem 2.2.1, we can conclude that $f_1(v,\xi(\omega)) \ge f_2(v,\xi(\omega))$ when the system cannot be drained before $T(\omega)$. Therefore, for the general case when we do not know the system is drained whether before recourse point, we can write the makespan function as follows:

$$
MS(v,\xi(\omega)) = \min\left\{f_1(v,\xi(\omega)), f_2(v,\xi(\omega))\right\}. \tag{4.4}
$$

Our goal in this chapter is to minimize the expected value of (4.4). In this setting, we can allow the accumulation of fluid in some buffers for some time, i.e., we do not require that incoming fluid be immediately drained until the time of the recourse action. Hence, we do not have a lower bound for the initial allocations (as in (2.9c)) and instead simply have nonnegativity constraints. Summarizing the developments above, we obtain the following formulation for the stochastic makespan problem with recourse:

$$\min_{v} \quad \mathbb{E}\left(MS(v,\tilde{\xi})\right) \tag{4.5a}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{C}_j} v_k \le 1, \qquad \forall j \in J \tag{4.5b}$$

$$v_k \ge 0, \ \ \forall k \in K. \tag{4.5c}$$

From Chapter 3, we know that $f_1(v,\xi)$ is convex in v for a fixed ξ. Since v is a parameter appearing on the right-hand side of the constraints (4.3b), $f_2(v,\xi)$ is also convex in v when ξ is fixed. However, $MS(v,\xi)$ is the minimum of two convex functions and is not, in general, convex. Hence, conventional methods of convex programming cannot be used directly to solve (4.5). So, we seek to exploit special structure in (4.4) to devise an algorithm to solve (4.5).

4.1 Solution Methodology

In this section, we develop a branch-and-bound method to solve model (4.5), i.e., the stochastic makespan problem with recourse. Norkin et al. [30] develop a general branch-and-bound algorithm for non-convex stochastic programs. We borrow some ideas from [30] and use the special structure of our problem wherever possible.

The idea behind our branch-and-bound algorithm is to partition the feasible region, so that within each partition the makespan function is convex for all realizations of the random parameters. For this purpose, we need a function $LB(v,\xi)$ that is convex with respect to v and acts as a lower bound for the original makespan function. Each node of the branch-and-bound tree, corresponds to a subset of the feasible region. If we minimize the expected value of $LB(v, \tilde{\xi})$ on this subset, we obtain a lower bound for the optimal expected makespan on this subset and a feasible allocation. This feasible allocation can be used to find an upper bound for the overall optimal expected makespan. Also, the function $LB(v, \xi)$ will be devised in such a way that if $MS(v, \xi)$ is convex on a node, i.e., on the subset of feasible allocations corresponding to the node, $LB(\cdot, \xi)$ will be exactly equal to the $MS(\cdot, \xi)$ on that node. In the following section, we first introduce a stochastic convex optimization problem which has these properties. Then, we discuss the ways to partition the feasible region.

4.1.1 A Lower Bound for the Recourse Problem

In this section, we develop a convex program whose optimal value is a lower bound on the optimal value of model (4.5). For this purpose, we first define a critical allocation level for each buffer and scenario. For a given scenario ω and a buffer k, the critical allocation level is defined as the allocation that drains the system exactly at time $T(\omega)$, if all the workload for buffer k in the system is present in the buffer at time 0. Hence, the critical allocation level is

$$v_k^c(\omega) = \frac{Q^k a(\omega) m_k(\omega)}{T(\omega)} + Q^k \alpha(\omega) m_k(\omega). \tag{4.6}$$

The following lemma gives a non-convex lower bounding function for the makespan function given in (4.4). We will use this lemma as a stepping stone to derive a convex lower bounding function.

Lemma 4.1.1. *Let v_k^c be as defined in (4.6) and*

$$M_k(\omega) = Q^k a(\omega) m_k(\omega) + Q^k \alpha(\omega) m_k(\omega) T(\omega).$$

Define

$$LB_k^1(v_k, \xi(\omega)) = \begin{cases} \dfrac{Q^k a(\omega) m_k(\omega)}{v_k - Q^k \alpha(\omega) m_k(\omega)} & \text{if } v_k \geq v_k^c(\omega) \\ \dfrac{M_k(\omega) - v_k T(\omega)}{1 - \sum_{l \in \mathcal{C}_{\sigma_k}} Q^l \alpha(\omega) m_l(\omega)} + T(\omega) & \text{if } v_k < v_k^c(\omega) \end{cases}. \tag{4.7}$$

Then

$$LB^1(v, \xi(\omega)) = \max_{k \in K} \left\{ LB_k^1(v_k, \xi(\omega)) \right\} \tag{4.8}$$

is a lower bound for function $MS(v, \xi(\omega))$ given in (4.4).

Proof. We will prove that each LB_k^1 is a lower bound for the draining time of buffer k. We consider three cases.

Case 1: $v_k \geq v_k^c(\xi(\omega))$ and the system drains before time $T(\omega)$. The function value is

$$LB_k^1(v_k, \xi(\omega)) = \frac{Q^k a(\omega) m_k(\omega)}{v_k - Q^k \alpha(\omega) m_k(\omega)},$$

and $MS(v, \xi(\omega))$ is the maximum of these functions over all buffers. Hence $LB^1(v, \xi(\omega)) = MS(v, \xi(\omega))$.

Case 2: $v_k \geq v_k^c(\xi(\omega))$ and the system drains after time $T(\omega)$. Then, $MS(v, \xi(\omega)) \geq T(\omega)$. Since, $v_k \geq v_k^c(\xi(\omega))$, we know that $LB_k^1(v_k, \xi(\omega)) \leq T(\omega) \leq MS(v, \xi(\omega))$.

Case 3: $v_k < v_k^c(\xi(\omega))$ and the system drains after time $T(\omega)$. In this case,

$$LB_k^1(v_k, \xi(\omega)) = \frac{M_k(\omega) - v_k T(\omega)}{1 - \sum_{l \in \mathcal{C}_{\sigma_k}} Q^l \alpha(\omega) m_l(\omega)} + T(\omega).$$

Let $x^*(\omega)$ be the optimal solution of (4.2) for scenario ω. Using (4.2b), it is easy to see

$$\begin{aligned} LB_k^1(v_k, \xi(\omega)) &\leq \frac{M_k(\omega) - x_k^*(\omega) m_k(\omega)}{1 - \sum_{l \in \mathcal{C}_{\sigma_k}} Q^l \alpha(\omega) m_l(\omega)} + T(\omega) \\ &\leq \frac{\sum_{l \in \mathcal{C}_{\sigma_k}} M_l(\omega) - x_l^*(\omega) m_l(\omega)}{1 - \sum_{l \in \mathcal{C}_{\sigma_k}} Q^l \alpha(\omega) m_l(\omega)} + T(\omega). \end{aligned}$$

To see why the second inequality holds, we move all the x_l's to left-hand side of (4.2c), and obtain the constraints (4.2c) in matrix form as

$$(I - P')x \leq a(\omega) + \alpha(\omega) T(\omega).$$

Multiplying both sides with Q, we see that the numerator in the first inequality is non-negative for all $k \in K$. Then we conclude that the second inequality holds, since all the summands are non-negative. The arguments above show that $LB_k^1(v_k, \xi(\omega))$ is a lower bound for $MS(v, \xi(\omega))$ for each k. Taking the maximum over all $k \in K$, we obtain the desired result. $\square$

If we are guaranteed to drain the system after time $T(\omega)$ for any feasible allocation, we only need to consider $f_2(v, \xi(\omega))$. Hence, the function is convex in v and there is no need for a lower bounding function. The function is similarly convex when the system is guaranteed to drain before time $T(\omega)$. We only require the lower bounding function for scenarios ω for which whether the system drains before $T(\omega)$ depends on the feasible allocation v. When the scenario ω is fixed, function $LB_k^1(v_k, \xi(\omega))$ is a univariate function in v_k and is piecewise convex, but of course, is non-convex. The function $LB_k^1(v_k, \xi(\omega))$ is convex in v_k when its derivative with respect to v_k is increasing, i.e., either $v_k^c(\omega) > 1$ or

$$-\frac{Q^k a(\omega) m_k(\omega)}{(v_k^c - Q^k \alpha(\omega) m_k(\omega))^2} \geq -\frac{T(\omega)}{1 - \sum_{l \in \mathcal{C}_{\sigma_k}} Q^l \alpha(\omega) m_l(\omega)}.$$

Substituting v_k^c's definition in the above inequality, its definition from equation (4.6), we obtain the following equivalent condition:

$$T(\omega)(1 - \sum_{l \in \mathcal{C}_{\sigma_k}} Q^l \alpha(\omega) m_l(\omega)) \leq Q^k a(\omega) m_k(\omega). \tag{4.9}$$

To be able to drain the system, the incoming workload should be drained as soon as it arrives in order to prevent accumulation of fluid in the system. The term in parenthesis on the left-hand side of equation (4.9) is the effort available to drain the initial workload at station σ_k. Either of these conditions, (4.9) and the condition $v_k^c(\omega) > 1$, means that the station cannot be drained before time $T(\omega)$. Hence, we can conclude that the function $LB_k^1(v_k, \xi(\omega))$ can be convex only when the station cannot be drained before time $T(\omega)$.

The above argument shows that the function $LB^1(v, \xi(\omega))$ is non-convex when the original makespan function is non-convex on the feasible region. However, by using the structure of $LB_k^1(v_k, \xi(\omega))$, it is possible to construct a convex lower bound. Figure 4.2 demonstrates an example of $LB_k^1(\cdot, \xi(\omega))$.

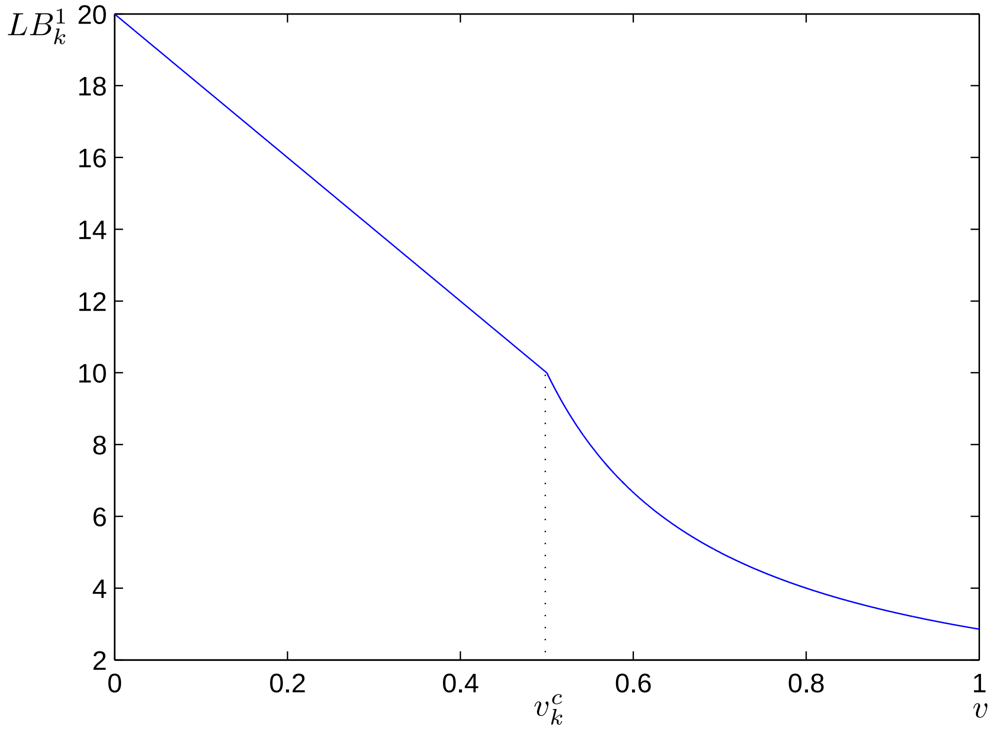

Figure 4.2: An instance of the non-convex lower bound function $LB_k^1(\cdot, \xi)$

Any feasible allocation in model (4.5), v, clearly lies in the hypercube $[0,1]^{|K|}$. The proposed branch-and-bound algorithm partitions this hypercube into smaller hypercubes on which the original makespan function is convex. Hence, at every node of the branch-and-bound algorithm, we only consider allocations in the hypercube of the form:

$$\Gamma = [v_1^L, v_1^U] \times \cdots \times [v_{|K|}^L, v_{|K|}^U]. \tag{4.10}$$

If for scenario ω, the system drains after time $T(\omega)$ for every allocation in Γ, we will simply use function $f_2(v, \xi(\omega))$. In the same manner, if for a scenario ω, the system drains before time $T(\omega)$ on Γ, we will use function $f_1(v, \xi(\omega))$. However, when the system drains before $T(\omega)$ for some $v \in \Gamma$ and after $T(\omega)$ for other $v \in \Gamma$, we require a convex function to serve as a lower bound for the makespan on Γ. To achieve this, we consider each $LB_k^1(v, \xi(\omega))$, and find the equation of the line which is tangent to $f_1(v, \xi(\omega))$ at some $\bar{v}_k > v_k^c$ and passes through the point $(v_k^L,\ LB_k^1(v, \xi(\omega)))$. Then, the function which takes the value on this line for each $v_k < \bar{v}_k$ and takes the value $\frac{Q^k a(\omega) m_k(\omega)}{v_k - Q^k \alpha(\omega) m_k(\omega)}$ for $v_k \geq \bar{v}_k$ will be our lower bound. Figure 4.3 illustrates this lower bound.

Now, we proceed with the derivation of the tangent line. At the tangent point $\bar{v}_k$, the slope of the line equals the derivative of $LB_k^1((v_k, \xi(\omega))$. Hence, the tangent point $\bar{v}_k$ satisfies the following equality

$$\frac{LB_k^1((\bar{v}_k, \xi(\omega)) - LB_k^1((v_k^L, \xi(\omega))}{\bar{v}_k - v_k^L} = -\frac{Q^k a(\omega) m_k(\omega)}{(\bar{v}_k - Q^k \alpha(\omega) m_k(\omega))^2}.$$

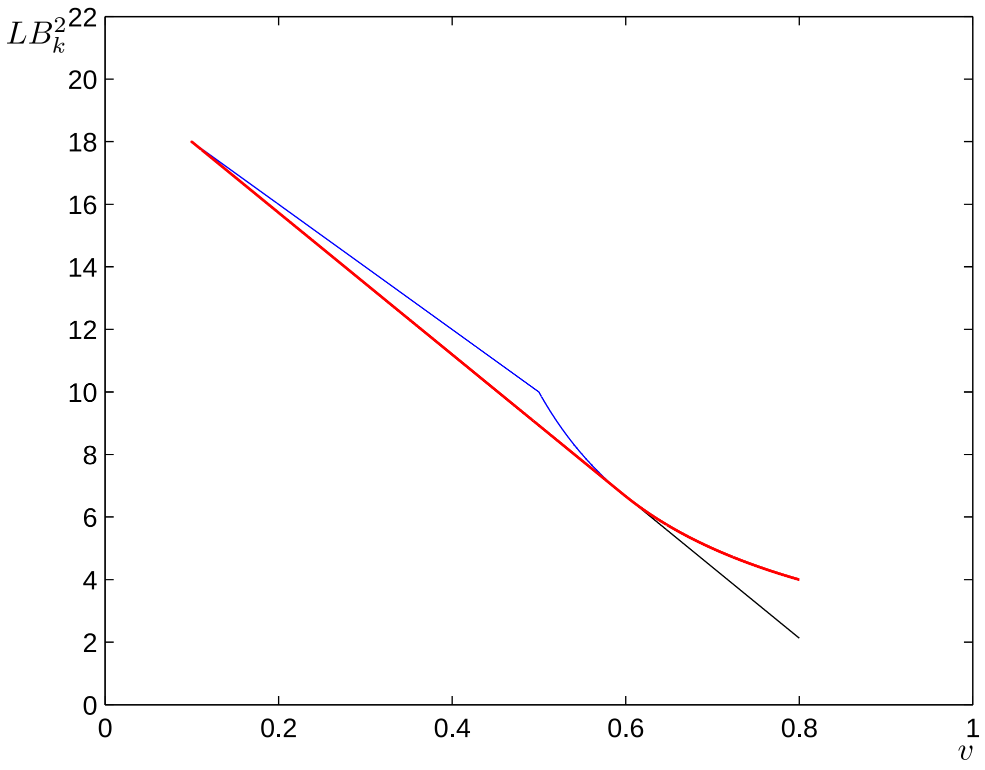

Figure 4.3: An instance of a convex lower bound function

Suppressing the ω's in the notation and solving this equation for $\bar{v}_k$, we obtain

$$\bar{v}_k = \frac{Q^k am_k + [(Q^k am_k)^2 + Q^k am_k LB_k^1(v_k^L, \xi)(Q^k \alpha m_k - v_k^L)]^{1/2}}{LB_k^1(v_k^L, \xi)} + Q^k \alpha m_k. \tag{4.11}$$

Since $LB_k^1(v_k^L, \xi(\omega)) < T(\omega)$, we know that $\bar{v}_k > v_k^c$. We are now ready to state the following lemma.

Lemma 4.1.2. *Let Γ be defined as in (4.10) and $\bar{v}_k$ be as in (4.11) and assume that (4.9) holds. Define*

$$LB_k^2(v, \xi(\omega)) = \begin{cases} \dfrac{Q^k a(\omega) m_k(\omega)}{v_k - Q^k \alpha(\omega) m_k(\omega)} & \text{if } v_k \geq \bar{v}_k(\omega) \\ -\dfrac{Q^k a(\omega) m_k(\omega)}{(\bar{v}_k - Q^k \alpha(\omega) m_k(\omega))^2}(v - v_k^L) + LB_k^1(v_k^L, \xi(\omega)) & \\ & \text{if } v_k < \bar{v}_k(\omega). \end{cases} \tag{4.12}$$

Then

$$LB^2(v, \xi(\omega)) = \max_{k \in K} \left\{ LB_k^2(v_k, \xi(\omega)) \right\} \tag{4.13}$$

is a lower bound for function $MS(v, \xi(\omega))$ given in (4.4) on the hypercube Γ. Also, $LB^2(v, \xi(\omega))$ is convex in v.

Proof. The fact that $LB_k^2(v_k, \xi(\omega)) \leq LB_k^1(v_k, \xi(\omega))$ is apparent from the above construction. Taking the maximum over $k \in K$ on both sides we see that $LB^2(v, \xi(\omega)) \leq LB^1(v, \xi(\omega))$; hence, it is also a lower bound for $MS(v, \xi(\omega))$.

To prove convexity, we observe that the derivative of $LB_k^2(v_k, \xi(\omega))$ with respect to v_k is increasing in v_k. Hence, $LB_k^2(v_k, \xi(\omega))$ is convex in v_k. Taking the maximum over all $k \in K$ preserves convexity. $\square$

Having derived a convex lower bound, we have the necessary tools to develop a branch-and-bound algorithm.

4.1.2 The Branch-and-Bound Algorithm

In our branch-and-bound algorithm, the purpose is to divide the feasible set into a finite number of subsets so that $\mathbb{E}(MS(v,\tilde{\xi}))$ is convex in each of these subsets. Then, it is possible to employ convex programming algorithms on each of these subsets to find the local optimum. After all local optima are found, the one with the smallest expected makespan yields the global optimum. Key to such an algorithm is the ability to demonstrate that certain subsets cannot contain the optimal solution. This can be ascertained using our lower bounds, which we further develop below, coupled with an incumbent feasible solution, i.e., the feasible solution with the best value of $\mathbb{E}(MS(v,\tilde{\xi}))$ obtained so far.To eliminate a hypercube from consideration it suffices to show the minimum value of the lower bound on the hypercube is at least that of the smallest upper bound found over all hypercubes. Finding an upper bound is the easier task, since any allocation $\hat{v}$ that satisfies the constraints (4.5b) and (4.5c) is suboptimal. Hence, $\mathbb{E}(MS(\hat{v},\tilde{\xi}))$ will yield an upper bound for the optimal expected makespan.

To find a lower bound for the optimal expected makespan, we make use of the lower bound in Lemma 4.1.2. Let Γ be as defined in (4.10). We classify scenario ω as *Type 1* on the hypercube Γ if

$$\frac{Q^k a(\omega) m_k(\omega)}{v_k^L - Q^k \alpha(\omega) m_k(\omega)} \leq T(\omega), \forall k \in K. \tag{4.14}$$

This condition states that even if we allocate the least effort possible on Γ to all buffers, all the buffers will still drain by time $T(\omega)$. Thus, for any feasible allocation $v \in \Gamma$, the system will be drained before time $T(\omega)$. In the same manner, we classify scenario ω as *Type 2* on the hypercube Γ, if there exists a buffer $k' \in K$ such that

$$\frac{Q^{k'}a(\omega)m_{k'}(\omega)}{v_{k'}^U - Q^{k'}\alpha(\omega)m_{k'}(\omega)} > T(\omega). \tag{4.15}$$

Hence, for scenario ω there is no feasible allocation $v \in \Gamma$ that drains the buffer k' before time $T(\omega)$. If neither of these conditions hold for scenario ω, then we say that ω is *Type 0* on Γ. We are now ready to state the convex lower bounding function that will be used in our branch-and-bound algorithm.

Theorem 4.1.3. *Let $LB(v, \xi(\omega)) : \Gamma \to \mathbb{R}$ and*

$$LB(v,\xi(\omega)) = \left\{ \begin{array}{cl} LB^2(v,\xi(\omega)) & \textit{if } \omega \textit{ is Type 0 on } \Gamma \\ f_1(v,\xi(\omega)) & \textit{if } \omega \textit{ is Type 1 on } \Gamma \\ f_2(v,\xi(\omega)) & \textit{if } \omega \textit{ is Type 2 on } \Gamma \end{array} \right. . \tag{4.16}$$

Then, $\mathbb{E}(LB(v,\tilde{\xi})) \leq \mathbb{E}(MS(v,\tilde{\xi}))$ for all $v \in \Gamma$ and $\mathbb{E}(LB(v,\tilde{\xi}))$ is convex on Γ.

Proof. We consider each scenario $\omega \in \Omega$ separately. It directly follows from Lemma 4.1.2 that if scenario ω is of Type 0 on Γ then $LB^2(v,\xi(\omega))$ is a lower bound for $MS(v,\xi(\omega))$. If scenario ω is Type 1 on Γ, then

$$f_1(v,\xi(\omega)) \leq T(\omega) \leq f_2(v,\xi(\omega)).$$

Hence, from its definition, we conclude $MS(v,\xi(\omega)) = f_1(v,\xi(\omega))$.

Now suppose, a scenario ω is of Type 2 and there exists a $v \in \Gamma$ such that $MS(v, \xi(\omega)) < f_2(v, \xi(\omega))$. However, $f_2(v, \xi(\omega))$ uses Theorem 2.2.1 to drain the remaining workload at time $T(\omega)$. Hence, if the system is not drained by time $T(\omega)$ using allocation vector v, it is not possible to drain the remaining workload in less than $f_2(v, \xi(\omega)) - T(\omega)$ time units.

We have proven that if $v \in \Gamma$, $LB(v, \xi(\omega)) \leq MS(v, \xi(\omega))$ for all $\omega \in \Omega$, hence $\mathbb{E}(LB(v, \tilde{\xi})) \leq MS(v, \tilde{\xi})$. In the same manner, $LB(v, \xi(\omega))$ is convex on Γ for all $\omega \in \Omega$. Since, expectation preserves convexity the result follows. $\square$

Above we introduced a way to classify scenarios. If all scenarios are classified as either Type 1 or Type 2 on a hypercube Γ, the lower bound is exactly equal to the expected makespan on Γ. Hence, when we are branching, the goal will be to decrease the number of scenarios that are classified as Type 0. If (4.15) holds for any buffer that is enough to characterize the scenario as Type 2 on the corresponding hypercube. However, to be able to classify a scenario as Type 1, (4.14) must be satisfied for all buffers in the system. Hence, there will be cases, where (4.14) is satisfied for some buffers, but the scenario is still classified as Type 0. If this is true for scenario ω, it is impossible to classify the scenario as Type 1 or Type 2 by branching on an allocation corresponding to a buffer that already satisfies (4.14). This suggests that in designing the algorithm, it will be beneficial to keep track of whether a buffer satisfies (4.14) for a specific scenario.

We say that a scenario-buffer pair (ω, k) is classified as Type 1, if (4.14) is satisfied for that k. The definition of Type 2 is unchanged, i.e., a scenario-buffer pair (ω, k) is classified as Type 2, if for 2 if for ω, (4.15) holds for any buffer $k' \in K$. For all other cases, the pair is classified as Type 0.

One of the key issues in the branch-and-bound algorithm is how to choose the branching variable. A natural way to choose the branching variable is to choose $k' \in K$ which has the highest number of scenarios (ω, k') classified as Type 0. Once a buffer k is chosen as the branching variable, then the problem is to find the branching value, i.e., the value in $[v_k^L, v_k^U]$ at which the hypercube is split into two. A natural candidate is the midpoint of the interval, but this is a naive method and ignores the structural properties of the objective function. Analyzing (4.14) and (4.15), we can see that a scenario-buffer pair (ω, k) is classified as Type 0 on Γ, if and only if

$$v_k^L < v_k^c(\omega) < v_k^U .$$

Hence, we can use $v_k^c(\omega)$ to determine the value at which we split the hypercube. Possible candidates for the branching value are the expected value and median of v_k^c.

To evaluate $\mathbb{E}(LB(v, \tilde{\xi}))$, we need to consider each scenario separately. Because of the computational issues, we assume that the random vector $\tilde{\xi}$ has finite support in the remainder of this section. Now, we have the tools to construct the branch-and-bound algorithm.

Input. The network structure (P, $\mathcal{C}_j$ for all $j \in J$), distributions of parameters a, α, m and T, the tolerance level ϵ

Output. An ϵ-optimal allocation vector

Step 0: Initialization. Set initial partition as $\Gamma_1 = [0,1]^{|K|}$ and the upper bound as $U = \infty$. Set $n = 1$, where n represents the current node. The current node has no children. Set $\delta = \epsilon/2$.

Step 1: Optimization. Define $LB : \Gamma_n \times \Xi \to \mathbb{R}$ via (4.16) and solve the current optimization problem up to δ-optimality

$$\begin{aligned} \min_{v} \quad & \mathbb{E}\left(LB(v,\tilde{\xi})\right) && (4.17a)\\ \text{s.t.} \quad & \sum_{k\in\mathcal{C}_j} v_k \le 1, \qquad \forall j \in J && (4.17b)\\ & v \in \Gamma_n. && (4.17c) \end{aligned}$$

Find a δ-optimal solution v^* and $z^* = \mathbb{E}(LB(v^*,\tilde{\xi}))$. If $z^* \ge U - \delta$, then go to Step 4. If $\mathbb{E}(MS(v^*,\tilde{\xi})) < U$, set $U = \mathbb{E}(MS(v^*,\tilde{\xi}))$ and go to Step 2.

Step 2: Convexity Check. If there are still scenarios classified as Type 0 on Γ_n, then go to Step 3 otherwise go to Step 4.

Step 3: Branching. Choose as the branching variable a $v_{k'}$ with the largest number of (ω, k') pairs classified as Type 0. Define $v^L_{k'} = \min\{v_{k'}|v \in \Gamma_n\}$ and $v^U_{k'} = \max\{v_{k'}|v \in \Gamma_n\}$, and choose the branching value $v^B_{k'n} = \mathbb{E}(\tilde{v}^c_{k'}|v^L_{k'} < \tilde{v}^c_{k'} < v^U_{k'})$. Set $n = n+1$ and form a child node to Node $(n-1)$ as the n^{th} node. Also set

$$\Gamma_n = \big([0,1] \times \cdots \times [v^L_{k'}, v^B_{k'n}] \times \cdots \times [0,1]\big) \cap \Gamma_{n-1}.$$

Go to Step 1.

Step 4: Fathoming. Trace the parent nodes of current node n backwards to find the first parent n' who has only one child. If there are no parents who have only one child go to Step 5. Otherwise, update $n = n + 1$ and form a child node to Node n' as the n^{th} node in the algorithm. Also set

$$\Gamma_n = \big([0,1] \times \cdots \times [v^B_{k'n'}, v^U_{k'n'}] \times \cdots \times [0,1]\big) \cap \Gamma_{n'}.$$

Then go to Step 1.

Step 5: Stop. The expected makespan U and associated solution v^* solve (4.5).

Each node of the branch-and-bound tree is fathomed either when the solutions corresponding to the node is provably suboptimal or when we reach a region where the original makespan is convex. When the original makespan function is convex for a node, the $LB(\cdot,\cdot)$ function is exactly equal to $MS(\cdot,\cdot)$. During the Optimization Step, we can employ any conventional convex nonlinear programming algorithm to minimize the lower bounding function. Using such a convex nonlinear programming algorithm, we can obtain a δ-optimal solution over the current hypercube. Hence, we can conclude that the solution obtained via the branch-and-bound algorithm also yields an ϵ-optimal expected makespan solution.

4.2 Computational Results

In this section, our goal is to assess the computational efficiency of the branch-and-bound algorithm described in Section 4.1.2. The Optimization

Step uses a convex nonlinear programming algorithm to optimize the lower bounding function at every node. As in Chapter 3, our choice will be Kelley's cutting-plane algorithm [26], which was briefly described in Section 3.3. This choice is again due to the nondifferentiability of the objective function.

When we were stating the branch-and-bound algorithm, we assumed that the support of the random variables is finite. When the support is large, or infinite, we can employ approximation methods. In Section 3.3, we discussed two approximation methods: one based on Monte Carlo sampling and the other on deterministically-valid bounds. The deterministically valid bounds rely on the objective function being convex in the random parameters. However, when the recourse action is present, the makespan function is the minimum of two functions, and this structure is destroyed, i.e., the objective function is not convex in the random parameters. As a result, the deterministic bounds stated in Section 3.3 are not valid for this chapter's model. The Monte Carlo sampling approximation is still valid and is used in this section.

As stated in Section 4.1.2, the convex nonlinear programming algorithms yields solutions within a pre-specified tolerance level ϵ. Our first goal is to test how the branch-and-bound method performs for difference tolerance levels. To assess the effect of the tolerance level, we tested our algorithm on a small reentrant line with 3 stations and 6 buffers. We assume that only one of the parameter vectors is random, i.e., only one of a, α or m is random and the recourse point T is deterministic. The random parameters are assumed to be independent and follow a uniform distribution. The data used for the networks in this section is given in Appendix 1. The distribution is approximated via samples of size $50, 100, 150, 200$ and 250. The solution times are given in Tables 4.1, 4.2 and 4.3.

Table 4.1: Computation times (seconds) for a reentrant line of 3 stations and 6 buffers for different tolerance levels and sample sizes when initial inventory is random

N	$\epsilon = 10^{-2}$	$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$
50	10.6004	27.5448	28.0687	30.5324
100	46.4159	30.6823	31.3922	31.5492
150	75.8055	129.831	120.461	135.069
200	155.91	176.548	173.926	152.212
250	27.2749	238.296	322.213	353.508

Table 4.2: Computation times (seconds) for a reentrant line of 3 stations and 6 buffers for different tolerance levels and sample sizes when incoming rate is random

N	$\epsilon = 10^{-2}$	$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$
50	37.8702	20.2529	17.9763	21.7027
100	76.1194	98.467	103.926	110.06
150	92.195	84.1242	98.1491	102.502
200	247.202	155.016	171.513	199.091
250	353.166	155.969	198.47	213.705

Decreasing the tolerance level ϵ, increases the solution time of the convex programs at each node. Since a convex program is solved at every node, one expects that the solution time of the branch-and-bound algorithm to increase as the tolerance level decreases. However, the computational studies

Table 4.3: Computation times (seconds) for a reentrant line of 3 stations and 6 buffers for different tolerance levels and sample sizes when service time is random

N	$\epsilon = 10^{-2}$	$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$
50	12.3261	25.8141	19.732	26.62
100	52.529	47.3468	53.7988	51.3362
150	167.924	73.2819	99.8648	114.357
200	207.431	113.083	116.658	121.473
250	73.4038	75.1056	93.5528	125.167

show that this is not always the case. This situation is due to the fathoming scheme. A node is fathomed if the lower bound for the node is greater than the upper bound for the optimal expected makespan. Decreasing the tolerance level yields stronger lower bounds at each node. Since the lower bounds increase, some nodes are fathomed earlier for tighter tolerance levels. In return, we have to solve less convex programs in the branch-and-bound algorithm, which decreases the overall computation time.

Another observation is that the solution time does not always increase with the sample size. Increasing the sample size results in more function evaluations in the algorithm, especially during the optimization of the convex lower bounds. Also the number of nodes is expected to increase as sample size increases. On the contrary, we see that sometimes increasing the sample size yields lower solution times. For example, in Table 4.3, for $\epsilon = 10^{-2}$ when there are 200 sample points the solution time is 207 seconds. When we add 50 more samples and solve a 250-scenario problem, we see that solution time decreases to 73 seconds. A detailed investigation on the algorithm shows that adding samples changes the branching variables and branching points considerably. As a result, some nodes are fathomed earlier than the 200 scenario problem, which decreases the solution time.

Summarizing our observations from Tables 4.1, 4.2 and 4.3, we see that a small problem (3 stations and 6 buffers) can be solved accurately ($\epsilon = 10^{-2}$) for moderate sample sizes ($N = 250$) around 5 minutes. Where we observe randomness plays a minor role in the solution time, i.e., solution times for random a, random α and random m are similar.

Another important issue that affects the solution time is the network size. Increasing the number of buffers increases the number of decision variables as well as the dimension of the parameters and also is expected to increase the number of nodes in the branch-and-bound tree. Increasing the number of stations increases the number of constraints at convex programs. Next, we test our branch-and-bound algorithm on networks of different sizes. The problem instances are solved to two significant digits, i.e., $\epsilon = 10^{-2}$.

In Tables 4.4, 4.5 and 4.6, we see that solution times increases considerably as we increase the size of the network. Above 20 buffers it becomes more difficult to solve problem instances. We see that when a is random and there are 250 scenarios the computation time is around 6 hours. Examining Tables 4.5 and 4.6 with random α and m, we see that this is the worst of the running times, but it is indicative of what can occur when using a branch-and-bound algorithm, in the worst case.

Table 4.4: Computation times (seconds) for different reentrant lines and different sample sizes when initial inventory is random

N	$3-6$	$2-10$	$5-10$	$5-20$
50	10.6004	74.8656	67.4078	1635.19
100	46.4159	157.38	62.1476	8965.37
150	75.8055	426.364	167.543	13157.4
200	155.91	734.503	623.919	15966.7
250	27.2749	1270.94	781.97	23248.6

Table 4.5: Computation times (seconds) for different reentrant lines and different sample sizes when incoming rate is random

N	$3-6$	$2-10$	$5-10$	$5-20$
50	23.7174	260.696	58.0852	555.112
100	124.95	443.785	135.74	1171.36
150	126.922	675.5	393.916	1663.09
200	188.854	865.401	611.809	2401.31
250	223.39	987.836	1171.01	3096.04

On the other hand, contrary to our intuition, we see that often the network with 5 stations and 10 buffers had a lower computation time than the network with 2 stations and 10 buffers. After a detailed investigation of the results, we see that having more buffers in a station enables us to classify scenarios earlier. This results in less nodes being evaluated in the algorithm.

Up to this point, we have assumed that at each node of the branch-and-bound tree, the convex programs are solved from scratch. Since solving convex programs plays a major role in the computational effort of our branch-

Table 4.6: Computation times (seconds) for different reentrant lines and different sample sizes when service time is random

N	$3-6$	$2-10$	$5-10$	$5-20$
50	23.0645	111.458	39.613	1361.35
100	56.8144	265.246	335.865	4821.12
150	104.413	540.05	3961.58	7850.28
200	128.519	641.598	277.167	11319.1
250	115.568	641.761	854.698	8116.25

and-bound algorithm, we can speed up the algorithm considerably if we solve these convex programs more quickly. The following lemma allows us to use Kelley's cutting-plane more efficiently in the branch-and-bound algorithm.

Lemma 4.2.1. *Let node n be a child node of node m and let $LB[n](v,\xi)$ be the lower bounding function used at node n. Then for any scenario $\omega \in \Omega$*

$$LB[n](v,\xi(\omega)) \geq LB[m](v,\xi(\omega)), \forall v \in \Gamma_n.$$

Proof. If ω is of the same type for both nodes, then it is obvious that

$$LB[n](v,\xi(\omega)) = LB[m](v,\xi(\omega))$$

for all $v \in \Gamma_n$. If ω is not of the same type for both nodes, it is classified as Type 0 for node m and it is classified as either Type 1 or Type 2 for node n. Since $\Gamma_n \subset \Gamma_m$, Lemma 4.1.2 states that

$$LB^2(v,\xi(\omega)) \leq \min\{f_1(v,\xi(\omega)), f_2(v,\xi(\omega))\}$$

for all $v \in \Gamma_n$. Hence, the result follows. $\square$

In Kelley's cutting-plane algorithm, the cut generated at each iteration acts as a lower bound for the objective function. Lemma 4.2.1 ensures us that the cuts generated at a node m are valid for all the child nodes of m. This enables us to solve convex programs for the child nodes more quickly.

Table 4.7: Computation times (seconds) for different reentrant lines and different sample sizes when initial inventory is random and cuts are inherited

N	$5-10$	$5-20$	$6-30$
50	17.1204	218.4	2091.62
100	21.1328	241.742	5090.24
150	43.0175	399.912	4284.79
200	94.2727	1052.86	10966
250	58.7151	583.529	5069.27

Table 4.8: Computation times (seconds) for different reentrant lines and different sample sizes when incoming rate is random and cuts are inherited

N	$5-10$	$5-20$	$6-30$
50	18.0743	201.916	92.299
100	35.4076	241.668	415.785
150	201.571	339.322	340.792
200	224.928	605.869	827.08
250	265.266	670.686	1024.18

Tables 4.7, 4.8 and 4.9 show the computational times when cuts are inherited by the child nodes in this manner. Contrasting the first column

Table 4.9: Computation times (seconds) for different reentrant lines and different sample sizes when service time is random and cuts are inherited

N	$5-10$	$5-20$	$6-30$
50	24.4583	221.131	167.993
100	37.1903	407.181	560.03
150	64.8281	425.445	1094.94
200	93.6928	452.09	2363.57
250	164.516	593.858	1389.18

of these tables with the values in Tables 4.4, 4.5 and 4.6, we see that it is possible to speed the algorithm up to 15 times with inheriting the cuts. With this scheme, we also see that it is possible to solve larger problems within reasonable computational times. In the remainder of the chapter, the results use the algorithm in which the child nodes inherit cuts from their parents.

As mentioned above, we have chosen T as indicated in Appendix 1. In choosing T we have aimed to form challenging test problems, i.e., so that the scenarios are classified by type as late as possible in the branch-and-bound algorithm. We illustrate this by solving the 5-station 10-buffer reentrant line with 250 scenarios for a range of values of T. By contrasting Table 4.10 with the values in Tables 4.7, 4.8 and 4.9, we see that our choices of T's in the above problems yield challenging problems.

Table 4.10 shows that as T grows the computation time tends to first grow and then shrink. When T is small, most of the scenarios are classified as Type 2 at the first node. Hence, as T increases the number of scenarios

Table 4.10: Computation times (seconds) for a reentrant line with 5 stations and 10 buffers and different recourse points

T	a random	α random	m random
15	14.3438	1.41079	3.47147
30	13.289	3.54746	21.7917
45	37.0004	5.13222	28.0987
60	58.7151	4.96524	25.8681
75	42.8295	95.4895	110.106
90	45.1411	265.266	134.724
105	34.4248	61.3657	36.2235
120	25.7071	59.9199	115.285
135	14.4858	60.3638	66.9598
150	6.28904	52.9879	82.6174
165	17.9903	70.2073	59.5609
180	15.2397	42.0896	44.3843
195	3.51746	28.4457	28.2137
210	2.35264	51.1582	20.7628
225	5.96909	29.9764	20.2889

classified as Type 0 increases. As a result the algorithm needs more nodes to find the region where original makespan is convex. However, for large T values, the scenarios are classified as Type 1 at the initial node and the computation time decreases.

So far, we have assume that the random parameters follow a uniform distribution. We also test our algorithm to see how the distribution of the parameters affects the computation time. For this purpose, we use a triangular distribution with the same support as the uniform distribution used in the previous computations. The network we use for this study is a reentrant line with 5 stations and 20 buffers. We also assume that the recourse point T is deterministic.

Table 4.11: Computation times (seconds) for a reentrant line with 5 stations and 20 buffers when parameters follow a triangular distribution

N	a random	α random	m random
50	101.69	130.212	196.087
100	404.148	182.937	230.03
150	296.082	247.545	209.707
200	428.516	322.18	312.224
250	265.279	379.978	239.652

Comparing Table 4.11 with the corresponding values in Tables 4.7, 4.8 and 4.9, we observe that the computation times tend to decrease when the parameters follow a triangular distribution. For uniform distribution, the sample is more dispersed on the support. Hence, the critical allocations are also dispersed and the algorithm needs to branch more to find the regions where the expected makespan is convex.

The branch-and-bound algorithm allows us to solve problems when the recourse point T is random. So far, we have assumed that T is deterministic. Now, we test our algorithm on a 5-station 10-buffer reentrant line when T is random. We assume that only one parameter vector is random and the parameters are independent and distributed uniformly. We use 250 random points to assess the efficiency of the branch-and-bound algorithm.

Table 4.12 and Table 4.13 shows the computation times when T follows

Table 4.12: Computation times (seconds) for a reentrant line with 5 stations and 10 buffers when the recourse point T follows a uniform distribution

N	a random	α random	m random
50	7.95679	36.7824	19.711
100	33.8699	44.6542	42.9465
150	10.9863	111.73	65.2011
200	66.6369	72.457	106.068
250	44.3863	296.025	110.398

Table 4.13: Computation times (seconds) for a reentrant line with 5 stations and 10 buffers when the recourse point T follows a triangular distribution

N	a random	α random	m random
50	11.8252	23.2425	24.4433
100	10.3304	40.5118	48.2157
150	23.9334	105.754	10.7014
200	33.179	116.123	96.2464
250	56.0165	59.492	95.4415

a uniform and a triangular distribution respectively. The computation times are not so sensitive to the distribution of T. However, when the results are compared with the results in Tables 4.7, 4.8 and 4.9, we generally see a decrease in computation times when T is random. In constructing Tables 4.7, 4.8 and 4.9, the recourse points are chosen to be around the optimal expected makespan. When we assume randomness, T deviates from these "worst-cases" allowing us to observe smaller computation times.

Chapter 5

Special Cases

In Chapter 3, we discussed examples showing that the solution obtained by using the expected values of the random parameters need not be optimal for the stochastic makespan problem. In general, it is not possible to state the solution of the stochastic problem analytically. That said, the purpose of this section is to describe special cases where it is possible to characterize the solution analytically.

5.1 Special Cases for Stochastic Makespan Problem without Recourse

5.1.1 Deterministic Station

While the expected value solution does not, in general, solve the stochastic problem, we can ask: If we have a stochastic fluid system, in which one of the stations is "deterministic," is it possible to say anything about the solution? The following theorem answers this question.

Theorem 5.1.1. *If $j^* \in J$ satisfies $Q^k a^{\omega_1} m_k^{\omega_1} = Q^k a^{\omega_2} m_k^{\omega_2} \equiv \beta_k$ and $Q^k \alpha^{\omega_1} m_k^{\omega_1} = Q^k \alpha^{\omega_2} m_k^{\omega_2} \equiv \rho_k$, $\forall \omega_1, \omega_2 \in \Omega$ and $\forall k \in \mathcal{C}_{j^*}$, then there is an optimal solution,*

v^*, to the stochastic makespan problem with

$$v_k^* = \frac{\beta_k - \beta_k \sum_{l \in \mathcal{C}_{j^*}} \rho_l + \sum_{l \in \mathcal{C}_{j^*}} \beta_l \rho_k}{\sum_{l \in \mathcal{C}_{j^*}} \beta_l}, \quad \forall k \in \mathcal{C}_{j^*}. \tag{5.1}$$

Proof. We first note that v_k^*, $k \in \mathcal{C}_{j^*}$, satisfies (3.1c) because (5.1)'s numerator is positive by condition (a) of Theorem 3.2.1, and (3.1b) for $j = j^*$ holds since

$$\sum_{k \in \mathcal{C}_{j^*}} v_k^* = \frac{\sum_{k \in \mathcal{C}_{j^*}} \beta_k - (\sum_{k \in \mathcal{C}_{j^*}} \beta_k)(\sum_{l \in \mathcal{C}_{j^*}} \rho_l) + (\sum_{l \in \mathcal{C}_{j^*}} \beta_l)(\sum_{k \in \mathcal{C}_{j^*}} \rho_k)}{\sum_{l \in \mathcal{C}_{j^*}} \beta_l} = 1.$$

Let $h_k(v_k)$ denote the draining time of buffer $k \in \mathcal{C}_{j^*}$, i.e., the right-hand side of equation (2.6). The draining time of the last buffer at station j^* is $\max_{k \in \mathcal{C}_{j^*}} h_k(v_k)$. Note that h_k is a decreasing function over feasible allocations and $h_k(v_k) = \frac{\sum_{l \in \mathcal{C}_{j^*}} \beta_l}{1 - \sum_{l \in \mathcal{C}_{j^*}} \rho_l}$, i.e., the draining time is equal for all buffers $k \in \mathcal{C}_{j^*}$. This coupled with $\sum_{k \in \mathcal{C}_{j^*}} v_k^* = 1$ implies

$$\max_{k \in \mathcal{C}_{j^*}} \{h_k(v_k^*)\} \leq \max_{k \in \mathcal{C}_{j^*}} \{h_k(v_k)\} \tag{5.2}$$

for all feasible allocations $v_k, k \in \mathcal{C}_{j^*}$. Suppose v^{**} solves the stochastic makespan problem, and extend v_k^* from (5.1) to $v_k^* = v_k^{**}, k \in K \setminus \mathcal{C}_{j^*}$. From (5.2), we have

$$\max_{k \in \mathcal{C}_{j^*}} \left\{ \frac{Q^k a^\omega m_k^\omega}{v_k^* - Q^k \alpha^\omega m_k^\omega} \right\} \leq \max_{k \in \mathcal{C}_{j^*}} \left\{ \frac{Q^k a^\omega m_k^\omega}{v_k^{**} - Q^k \alpha^\omega m_k^\omega} \right\}, \quad \forall \omega \in \Omega. \tag{5.3}$$

We know that v^* and v^{**} drain all other buffers at the same time, hence

$$\max_{k \in K} \left\{ \frac{Q^k a^\omega m_k^\omega}{v_k^* - Q^k \alpha^\omega m_k^\omega} \right\} \leq \max_{k \in K} \left\{ \frac{Q^k a^\omega m_k^\omega}{v_k^{**} - Q^k \alpha^\omega m_k^\omega} \right\}, \quad \forall \omega \in \Omega.$$

Taking expectations,

$$\mathbb{E}\left(\max_{k\in K}\left\{\frac{Q^k\tilde{a}\tilde{m}_k}{v_k^* - Q^k\tilde{\alpha}\tilde{m}_k}\right\}\right) \leq \mathbb{E}\left(\max_{k\in K}\left\{\frac{Q^k\tilde{a}\tilde{m}_k}{v_k^{**} - Q^k\tilde{\alpha}\tilde{m}_k}\right\}\right).$$

Hence, v^* also solves the stochastic makespan problem. $\square$

Theorem 5.1.1 implies that if the random parameters defining our stochastic makespan problem have a certain structure, then the expected value solution solves the stochastic problem. In the next three subsections, we clarify this implication by examining three other special, intuitive cases.

5.1.2 Random Incoming Rates

In this subsection, we assume that only the incoming rate vector α is random and that it has a special probabilistic structure. Specifically, we assume that randomness is observed proportionally for all buffers, i.e., there is a deterministic base rate vector α^0, and for any scenario, $\omega \in \Omega$, the rate vector can be represented as $N^\omega \alpha^0$. Here, N^ω is a scalar determined by scenario ω. This is equivalent to assuming that fluid arrives to the system from a single source with an unknown rate, but it is distributed to the stations in the system according to fixed proportions. Note that since fluid reentrant lines have exogenous arrivals to only one buffer, this structural assumption always holds for stochastic makespan problems in such networks.

With the assumption above, we can construct the following special case.

Theorem 5.1.2. *If $\tilde{\alpha} = \tilde{N}\alpha^0$ and $j^* \in J$ satisfies $\frac{Q^k\alpha^0}{Q^k a} = \frac{Q^l\alpha^0}{Q^l a}, \forall k, l \in \mathcal{C}_{j^*}$,* *then there is an optimal solution, v^*, to the stochastic makespan problem with*

$$v_k^* = \begin{cases} \frac{Q^k\alpha^0 m_k}{\sum_{l\in\mathcal{C}_{j^*}} Q^l\alpha^0 m_l}, & \textit{if } \sum_{l\in\mathcal{C}_{j^*}} Q^l\alpha^0 m_l > 0 \\ \frac{Q^k a m_k}{\sum_{l\in\mathcal{C}_{j^*}} Q^l a m_l}, & \textit{if } \sum_{l\in\mathcal{C}_{j^*}} Q^l\alpha^0 m_l = 0 \end{cases}, \forall k \in \mathcal{C}_{j^*}. \tag{5.4}$$

Proof. We first show that all the buffers in station j^* are drained at the same time for each scenario. If $\sum_{l\in\mathcal{C}_{j^*}} Q^l\alpha^0 m_l = 0$, then j^* satisfies the conditions of Theorem 5.1.1. Moreover, v_k^* from (5.4) is identical to that of (5.1), and hence, from the proof of Theorem 5.1.1 all the buffers at j^* are drained at the same time. If $\sum_{l\in\mathcal{C}_{j^*}} Q^l\alpha^0 m_l > 0$ then $\sum_{k\in\mathcal{C}_{j^*}} v_k^* = 1$ and, as before, (3.1c) holds by condition (a) of Theorem 3.2.1. Then, for any $k, l \in \mathcal{C}_{j^*}$ and $\omega \in \Omega$

$$\begin{aligned} \frac{Q^k a m_k}{\dfrac{Q^k\alpha^0 m_k}{\sum_{i\in\mathcal{C}_{j^*}} Q^i\alpha^0 m_i} - N^\omega Q^k\alpha^0 m_k} &= \frac{Q^k a m_k}{Q^k\alpha^0 m_k\left(\dfrac{1}{\sum_{i\in\mathcal{C}_{j^*}} Q^i\alpha^0 m_i} - N^\omega\right)} \\ &= \frac{Q^l a m_l}{Q^l\alpha^0 m_l\left(\dfrac{1}{\sum_{i\in\mathcal{C}_{j^*}} Q^i\alpha^0 m_i} - N^\omega\right)} \\ &= \frac{Q^l a m_l}{\dfrac{Q^l\alpha^0 m_l}{\sum_{i\in\mathcal{C}_{j^*}} Q^i\alpha^0 m_i} - N^\omega Q^l\alpha^0 m_l}. \end{aligned}$$

Hence, in each scenario, the proposed solution drains all the buffers at j^* at the same time.

Next, we show that v^* leads to a finite expected draining time for all buffers at j^*. We assume that conditions (a) and (b) for Theorem 3.2.1 are satisfied. Using (a), we know that $\sum_{k\in C_{j^*}} \operatorname{ess\,sup}\{\tilde{N}\} Q^k\alpha^0 m_k \leq 1$. Hence,

$$\operatorname{ess\,sup}\{\tilde{N}\} \leq 1/\sum_{k\in C_{j^*}} Q^k\alpha^0 m_k. \tag{5.5}$$

If the inequality holds strictly, there exists an $\epsilon > 0$, such that

$$E\left(\frac{Q^k a m_k}{v_k^* - \tilde{N} Q^k \alpha^0 m_k}\right) < \frac{Q^k a m_k}{\epsilon}, \forall k \in \mathcal{C}_{j^*}.$$

On the other hand if the inequality holds as an equality, then $v_k^* = S_k$, $\forall k \in \mathcal{C}_{j^*}$. Using (b) of Theorem 3.2.1 we conclude that v^* leads to a finite expected draining time for all buffers at j^*.

Suppose v^{**} solves the stochastic makespan problem, and extend the definition of v_k^* from (5.4) to $v_k^* = v_k^{**}, k \in K \setminus \mathcal{C}_{j^*}$. The proof can now be completed using the same argument as in the proof of Theorem 5.1.1. $\square$

Note that Theorem 5.1.2 holds even if there is a j^*, such that $\frac{Q^k \alpha^0}{Q^k a} = \frac{Q^l \alpha^0}{Q^l a} = \infty \ \forall k, l \in \mathcal{C}_{j^*}$, (i.e., $Q^k a = Q^l a = 0$). Since the necessary and sufficiency conditions are satisfied, (5.5) implies $\frac{Q^k \alpha^0 m_k}{\sum_{i \in \mathcal{C}_{j^*}} Q^i \alpha^0 m_i} - N^\omega Q^k \alpha^0 m_k \geq 0$, $\forall \omega \in \Omega$. Hence, the station stays empty for all scenarios and all the buffers are still drained at the same time, so the result follows.

5.1.3 Random Service Rates

In the previous subsection, the arrival rates for all buffers in the system were perfectly correlated. Since the arrival rates may be determined by the same causes in the exogenous environment and there are systems like reentrant lines, such a dependency assumption could naturally arise. However, assuming a similar structure for the system's service times may be overly restrictive. Fortunately, in the case where service rates are random, similar results hold with a relaxed version of the dependency assumption. In particular, we need only assume that service rates for buffers within the same station are proportional for all scenarios $\omega \in \Omega$. That is, there is a base service time m^0 and for any scenario ω, $m_k^\omega = N_j^\omega m_k^0$. Here, N_j^ω is determined by station j and scenario ω, and may differ by station under the same scenario. This probabilistic structure could arise as follows. Suppose there are several identical machines at each station with deterministically known service times, but the number of machines in working condition is unknown when the allocation policy must be specified. In this case, a fluid model with the random service rate structure above may serve as a reasonable approximation.

The next theorem allows us to present a result useful for systems in which a and α are not random, and the service rates are correlated in the manner discussed above.

Theorem 5.1.3. *If $j^* \in J$ satisfies $\tilde{m}_k = \tilde{N}_{j^*} m_k^0, k \in \mathcal{C}_{j^*}$ and $\frac{Q^k \alpha}{Q^k a} = \frac{Q^l \alpha}{Q^l a}$ where $k, l \in \mathcal{C}_{j^*}$, then there is an optimal solution, v^*, to the stochastic makespan problem with*

$$v_k^* = \begin{cases} \frac{Q^k \alpha m_k^0}{\sum_{l \in \mathcal{C}_{j^*}} Q^l \alpha m_l^0}, & \text{if } \sum_{l \in \mathcal{C}_{j^*}} Q^l \alpha m_l^0 > 0 \\ \frac{Q^k a m_k^0}{\sum_{l \in \mathcal{C}_{j^*}} Q^l a m_l^0}, & \text{if } \sum_{l \in \mathcal{C}_{j^*}} Q^l \alpha m_l^0 = 0 \end{cases}, \forall k \in \mathcal{C}_{j^*}. \qquad (5.6)$$

Proof. Using the same approach as in Theorem 5.1.2, it can be shown that all buffers at station j^* are drained at the same time for each scenario. Then the result follows from the argument used in the proofs of Theorems 5.1.1 and 5.1.2. □

5.1.4 Random Initial Inventory

As a final special case, we consider a system in which α and m are deterministic, but the initial inventory vector $\tilde{a}$ is random with perfectly correlated components.

Theorem 5.1.4. *If $\tilde{a} = \tilde{N}a^0$ then*

$$v_k^* = \frac{\beta_k - \beta_k \sum_{l \in \mathfrak{C}_{\sigma_k}} \rho_l + \sum_{l \in \mathfrak{C}_{\sigma_k}} \beta_l \rho_k}{\sum_{l \in \mathfrak{C}_{\sigma_k}} \beta_l}, \quad \forall k \in K, \tag{5.7}$$

where $\beta_k = Q^k a^0 m_k$ and $\rho_k = Q^k \alpha m_k$, solves both the expected value and stochastic makespan problems.

Proof. Notice that,

$$\mathbb{E}\left(\max_{k \in K} \left\{ \frac{Q^k \tilde{N} a^0 m_k}{v_k - Q^k \alpha m_k} \right\} \right) = \mathbb{E}(\tilde{N}) \max_{k \in K} \left\{ \frac{Q^k a^0 m_k}{v_k - Q^k \alpha m_k} \right\}.$$

Hence, minimizing $\max_{k \in K} \left\{ \frac{Q^k a^0 m_k}{v_k - Q^k \alpha m_k} \right\}$ subject to (3.1b) and (3.1c) yields an allocation that solves both the stochastic and expected value versions of the makespan problem. The form of v^* given in (5.7) then follows by applying Theorem 5.1.1. $\square$

5.2 Special Cases for Stochastic Makespan Problem with Recourse

In the previous section, we outlined some special cases where an analytic solution can be found for the stochastic makespan problem without recourse. Theorem 5.1.4 states that under certain assumptions on the distribution of the

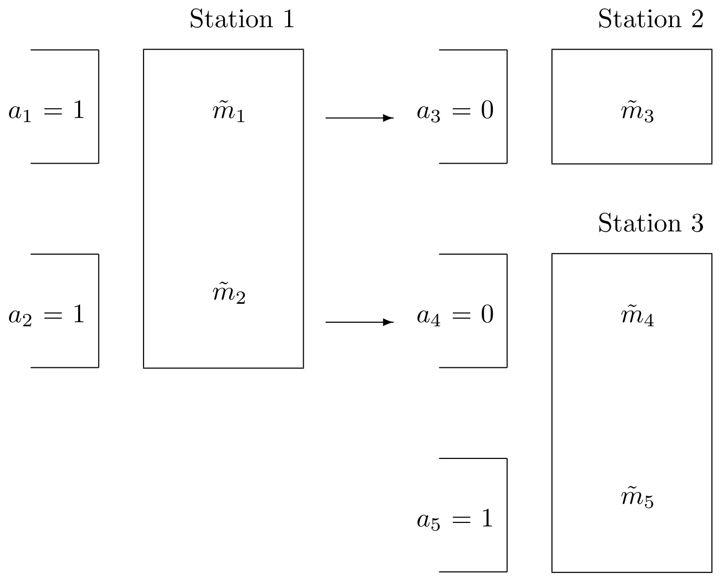

Figure 5.1: Network with Three Stations and Five Buffers

initial inventory, the suggested allocations drain the system in minimal time for all scenarios. Hence, this result is also valid when a recourse action is allowed. However, counterexamples can be found to show that Theorems 5.1.1, 5.1.2 and 5.1.3 do not hold in the recourse case.

Consider the 3-station 5-buffer network in Figure 5.1. Assume that the service time vector is random with $\mathbb{P}(\tilde{m} = (1, 3, 4, 2, 2)) = \mathbb{P}(\tilde{m} = (2, 1, 4, 2, 2)) = 0.5$ and there are no exogenous arrivals to the system. The initial inventory vector is also given by $a = (1, 1, 0, 0, 1)$. The initial allocations are set before the system starts running and the recourse action is allowed at $T = 2$. Under both scenarios, both stations 1 and 3 require 4 time units to drain their initial workload, hence $\mathbb{E}(MS(v, \tilde{\xi})) \geq 4$. If the allocations are set to

$v = (1, 0, 1, 0, 1)$, the system can be drained in 4 time units for all scenarios and this lower bound is achieved. On the other hand, station 3 satisfies the conditions stated in Theorem 5.1.1 and Theorem 5.1.3. If these theorems hold in the recourse case, there should be a solution with $v_4 = v_5 = 0.5$. However, if $\tilde{m} = (1, 3, 4, 2, 2)$, Station 1 is able to drain only 1/3 units of fluid from buffer 2 in unit time. As a result, the capacity allocated to buffer 4 is not fully utilized. This leads to a makespan strictly greater than 4 for this scenario. Hence, we conclude that any allocation where $v_4 = 0.5$ is suboptimal. A similar counterexample can be constructed when the incoming rate is random.

Even though the above counterexample shows that the theorems stated in Section 5.1 do not generally hold in the recourse case, we can prove that similar results hold under more restrictive assumptions. The remainder of this section is devoted to the special cases where the stochastic makespan problem with recourse is analytically solvable.

5.2.1 Random Incoming Rates

Theorem 5.1.2 states that if there exists a station satisfying the given conditions optimal allocations for that station can be determined by (5.4). It is possible to construct a similar counterexample demonstrating that this result is no longer valid for the recourse case. However, when all the stations satisfy the conditions in the theorem below, the result holds.

Theorem 5.2.1. *If $\tilde{\alpha} = \tilde{N}\alpha^0$ and for all stations $j \in J$, $\frac{Q^k \alpha^0}{Q^k a} = \frac{Q^l \alpha^0}{Q^l a}, \forall k, l \in \mathcal{C}_j$, then there is an optimal solution, v^*, to the stochastic makespan problem* *with*

$$v_k^* = \begin{cases} \frac{Q^k\alpha^0 m_k}{\sum_{l\in\mathcal{C}_{j^*}} Q^l\alpha^0 m_l}, & \textit{if } \sum_{l\in\mathcal{C}_{j^*}} Q^l\alpha^0 m_l > 0 \\ \frac{Q^k a m_k}{\sum_{l\in\mathcal{C}_{j^*}} Q^l a m_l}, & \textit{if } \sum_{l\in\mathcal{C}_{j^*}} Q^l\alpha^0 m_l = 0 \end{cases}, \forall k \in K. \tag{5.8}$$

Proof. When initial allocations are as given in (5.8),

$$\begin{aligned} \frac{Q^k a m_k}{\dfrac{Q^k\alpha^0 m_k}{\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i\alpha^0 m_i} - N^\omega Q^k\alpha^0 m_k} &= \frac{Q^k a m_k}{Q^k\alpha^0 m_k\left(\dfrac{1}{\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i\alpha^0 m_i} - N^\omega\right)} \\ &= \frac{Q^k a m_k \displaystyle\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i\alpha^0 m_i}{Q^k\alpha^0 m_k\left(1 - N^\omega \displaystyle\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i\alpha^0 m_i\right)}. \end{aligned}$$

Using the fact that $\frac{Q^k\alpha^0}{Q^k a} = \frac{\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i\alpha^0}{\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i a}$ and taking the maximum over all buffers, we get

$$\max_{k\in K}\left\{\frac{Q^k a m_k}{\dfrac{Q^k\alpha^0 m_k}{\displaystyle\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i\alpha^0 m_i} - N^\omega Q^k\alpha^0 m_k}\right\} = \max_{k\in K}\left\{\frac{\displaystyle\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i a m_i}{1 - N^\omega \displaystyle\sum_{i\in\mathcal{C}_{\sigma_k}} Q^i\alpha^0 m_i}\right\}.$$

Using Theorem 2.2.1, we conclude that the initial allocations (5.8) yield the minimum makespan possible for each scenario. Hence, no recourse action is needed and the expected makespan is minimized. □

5.2.2 Random Service Rates

We have shown that when recourse action is allowed Theorem 5.1.3 no longer holds. However, this theorem can be generalized for the recourse case under more restrictive assumptions as in Theorem 5.2.1.

Theorem 5.2.2. *If for all $j \in J$ $\tilde{m}_k = \tilde{N}_{j^*} m_k^0, \forall k \in \mathcal{C}_j$ and $\frac{Q^k \alpha}{Q^k a} = \frac{Q^l \alpha}{Q^l a}$ for all $k, l \in \mathcal{C}_j$, then there is an optimal solution, v^*, to the stochastic makespan problem with*

$$v_k^* = \begin{cases} \frac{Q^k \alpha m_k^0}{\sum_{l \in \mathcal{C}_{j^*}} Q^l \alpha m_l^0}, & \text{if } \sum_{l \in \mathcal{C}_{j^*}} Q^l \alpha m_l^0 > 0 \\ \frac{Q^k a m_k^0}{\sum_{l \in \mathcal{C}_{j^*}} Q^l a m_l^0}, & \text{if } \sum_{l \in \mathcal{C}_{j^*}} Q^l \alpha m_l^0 = 0 \end{cases}, \forall k \in K. \tag{5.9}$$

Proof. The proof is similar to the proof of Theorem 5.2.1. Using the same arguments, we show that the draining time under the suggested allocations is equal to the theoretical lower bound given in Theorem 2.2.1. Hence, the allocations are optimal. □

5.2.3 Non-idling Stations

The system drains when the total effort spent on draining each buffer is equal to the sum of the total workload initially present in the system and the workload arriving to the system till the draining time. Theorem 5.1.1 provides a lower bound which can be attained under policies when the stations do not idle as long as there is work present somewhere in the system. In this work, our main focus is on stationary policies, where some capacity is allocated for a buffer initially. This allocated capacity cannot be used for other buffers unless a recourse action is taken. Due to parameter uncertainty, this sometimes results in under-utilization of some stations. That is, after a buffer drains, some of the capacity allocated for that buffer is wasted.

This argument implies that if we fully utilize the capacity of stations till the recourse action, then we are guaranteed to obtain a smaller makespan than the makespan of a policy which may have idling. The next theorem follows as a result of this observation.

Theorem 5.2.3. *If for all $j \in J$, there exists a $k_j \in \mathcal{C}_j$, such that*

$$\frac{\tilde{a}_{k_j}\tilde{m}_{k_j}}{\tilde{\alpha}_{k_j}\tilde{m}_{k_j}} \geq \tilde{T} \text{ w.p.1} \tag{5.10}$$

then setting $v_{k_j} = 1$ for all $j \in J$ up to time $\tilde{T}$ is optimal.

Proof. Since, the system cannot be drained till $\tilde{T}$ w.p.1, $MS(v, \tilde{\xi}) = f_2(v, \tilde{\xi})$ almost surely. Using (5.10), (4.2b) and (4.2c), we conclude that setting $x_{k_j} = \tilde{T}/m_{k_j}$ solves (4.2). Hence,

$$f_2(v, \tilde{\xi}) = \max_{j \in J} \left\{ \frac{\sum_{k \in \mathcal{C}_j} Q^k(\tilde{a} + \tilde{\alpha}\tilde{T})\tilde{m}_k - \tilde{T}}{1 - \sum_{k \in \mathcal{C}_j} Q^k \tilde{\alpha}\tilde{m}_k} \right\} + \tilde{T},$$

which is also equal to the lower bound in (2.5). □

Chapter 6

Conclusions and Future Work

Multiclass queueing and fluid networks are useful tools for analyzing complex manufacturing environments. In real life, the parameters of the manufacturing system are not known deterministically. However, some strategic and control decisions should be made under the assumption of parameter uncertainty. To analyze how this uncertainty affects the decision making process, we focus on the makespan problem in fluid networks. The makespan of the system is defined as the time when the workload in the system is drained completely. The controller must allocate the capacity at each station to minimize the makespan. In this work, our goal is to analyze the structural properties of the makespan problem in the presence of parameter uncertainty and develop tools for optimization of the system using stochastic programming techniques.

In Chapter 2, we introduce the makespan problem and investigate the results in the literature. When the parameters of the system are known deterministically, it is possible to solve the problem analytically. However, to have a better understanding of the stochastic case, we give a mathematical programming model in this chapter. We also observe that the deterministic problem can be solved for each station separately.

In Chapter 3, we assume that the parameters, i.e., the initial inventory, exogenous arrival rates and the service times, are not known deterministically. The controller must decide the capacity allocations before the system begins operation and these allocations cannot be changed later on. We begin this chapter by contrasting the deterministic and stochastic problems. Using counterexamples, we show that the stochastic problem differs from the deterministic case considerably. We also show that satisfying the usual traffic conditions for all possible scenarios is not enough to guarantee the existence of a finite expected makespan. Hence, we derive necessary and sufficient conditions for the well-posedness of the problem. Next, we formulate the stochastic problem as a convex nonlinear stochastic program. We propose a solution methodology using cutting plane techniques along with Monte Carlo simulation and deterministic bounding techniques.

In Chapter 4, we allow the controller to take a recourse action after some random time. One motivation for this framework is a system in which the controller decides on the initial allocations and then performs a data collection process. When the data collection process is over after some deterministic or random time, the decision maker has deterministic information about the system parameters. After this point, the decision maker can base his decisions on the results of the deterministic makespan problem. However, his initial goal is to operate the system optimally during the data collection process. We formulate the problem as a stochastic nonlinear program. However, we observe that the objective function is non-convex in this case. Exploiting the special structure of the objective function, we develop a branch-and-bound methodology to solve the problem. The main idea in the algorithm is to partition the feasible region in order to obtain regions where the objective function is convex. Then, we perform a detailed computational study and see that it is possible to solve small and medium-size problems efficiently.

Our results in Chapter 3 indicate that in general it is not possible to derive an analytic solution to the makespan problem under parameter uncertainty. In Chapter 5, our goal is to identify cases where the problem is analytically solvable. We treat the problems in Chapter 3 and Chapter 4 separately. We state four special cases when no recourse action is allowed, in which the problem is solvable analytically. We also show that in the presence of a recourse action, the closed form solutions no longer hold. However, it is possible to prove that similar results hold in the recourse case under more restrictive assumptions. We also show that if we can guarantee that the system does not idle till the recourse action, then the optimal expected makespan is obtained.

In the current setting, we allow only one recourse action for the decision maker. One clear cut way to extend this problem is to allow the decision maker to take multiple recourse actions as the uncertainty in the parameters is gradually reduced. Another way to modify the decision structure is to allow the controller to select a processor sharing policy, rather than a policy based on a fixed allocation vector.

It is also clear that for some applications, the makespan objective is inappropriate. The deterministic fluid model with a linear holding cost objective has been widely studied in the literature. One contribution of this paper is to show that even the basic properties of the makespan problem change when the parameters are viewed as being random. Hence, the work herein raises the question of how the optimization characteristics of fluid problems under various objective functions change when random parameters are introduced.

Appendix

Appendix 1

The Data for Fluid Networks Used in Computations

In this work, we used reentrant lines as our test problems. Hence, for each network the entries in the routing matrix P can be characterized as follows:

$$p_{kl} = \begin{cases} 1 & \text{if } l = k+1 \\ 0 & \text{otherwise} \end{cases}.$$

For all the reentrant lines, the set of buffers which is processed by station j is given with the following expression:

$$\mathcal{C}_j = \{k \in K | j \equiv k \pmod{|J|} + 1\}.$$

The data given below is randomly generated.

1.1 3-Station 6-Buffer Reentrant Line

The initial inventory in this network is

$$a = (15, 14, 13, 12, 11, 10).$$

The incoming rate is given by

$$\alpha = (1, 0, 0, 0, 0, 0)$$

and the service time is

$$m = (0.3, 0.4, 0.3, 0.4, 0.3, 0.4).$$

If a parameter is random, then it is distributed uniformly between 0 and the value stated above. For this network the recourse action is taken at $T = 75$.

1.2 2-Station 10-Buffer Reentrant Line

For this network, we assume the initial inventory to be

$$a = (10, 8, 4, 2, 1, 1, 9, 4, 1, 4),$$

the incoming rate is

$$\alpha = (0.0047616, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

and the service time is

$$\begin{aligned} m = \ & (0.84913, 0.67874, 0.74313, 0.65548, 0.70605, 0.27692, 0.097132, 0.69483, \\ & 0.95022, 0.43874) \end{aligned}$$

As in 3-station and 6-buffer network, if a parameter is random, then it follows a uniform distribution between 0 and the value above. The recourse time T is 55 when initial inventory is random, 110 when incoming rate is random and 75 when service time is random.

1.3 5-Station 10-Buffer Reentrant Line

We start with the initial inventory

$$a = (2, 7, 3, 10, 2, 10, 9, 5, 8, 7),$$

the incoming rate vector is

$$\alpha = (0.27157, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

and the service time vector is

$$m = \begin{array}{l} (0.90579, 0.91338, 0.09754, 0.54688, 0.96489, 0.97059, 0.48538, 0.14189, \\ 0.91574, 0.95949) \end{array}$$

For this network, if a parameter is random and uniformly distributed, it is distributed between 0 and the value above. When the recourse time is deterministic it is assumed to be 90. If T is random, the support is $(70, 110)$ for both uniform and triangular distributions. Triangular distribution has its most likely value at 90.

1.4 5-Station 20-Buffer Reentrant Line

The initial inventory for this network is

$$a = (10, 8, 4, 2, 1, 1, 9, 4, 1, 4, 8, 5, 7, 8, 7, 2, 5, 4, 3, 3),$$

the incoming rate vector is

$$\alpha = (0.0059519, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

and the service time vector is

$$m = \begin{array}{l} (0.84913, 0.67874, 0.74313, 0.65548, 0.70605, 0.27692, 0.097132, 0.69483, \\ 0.95022, 0.43874, 0.76552, 0.18687, 0.44559, 0.70936, 0.27603, 0.6551, \\ 0.119, 0.95974, 0.58527, 0.75127). \end{array}$$

When a parameter is random, it is uniformly distributed between 0 and the values above. If the parameter follows a triangular distribution, then the support is the same as in the uniform case, having the most likely value at the midpoint of the interval. The recourse time T for a-random and m-random cases is 75, when α is random $T = 110$.

1.5 6-Station 30-Buffer Reentrant Line

The initial inventory vector is

$$\begin{aligned} a = \ & (9, 6, 2, 9, 9, 10, 2, 7, 4, 6, 10, 8, 4, 1, 6, 10, 6, 1, 2, 4, 2, 3, 7, 5, \\ & 3, 2, 6, 1, 2, 1), \end{aligned}$$

the incoming rate vector is

$$\begin{aligned} \alpha = \ & (0.067461, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \\ & 0, 0, 0, 0, 0, 0, 0, 0, 0), \end{aligned}$$

and the service time vector is

$$\begin{aligned} m = \ & (0.69908, 0.95929, 0.13862, 0.25751, 0.25428, 0.24352, 0.34998, 0.25108, \\ & 0.47329, 0.83083, 0.54972, 0.28584, 0.75373, 0.56782, 0.05395, 0.77917, \\ & 0.12991, 0.46939, 0.33712, 0.79428, 0.52853, 0.60198, 0.65408, 0.74815, \\ & 0.083821, 0.91334, 0.82582, 0.99613, 0.44268, 0.9619), \end{aligned}$$

As in the previous networks the parameters are assumed to be uniformly distributed between 0 and the values above. The recourse time T is 280 for a-random case, 450 for α-random case and 380 for m-random case.

Bibliography

[1] E. J. Anderson and P. Nash. *Linear Programming in Infinite Dimensional Spaces*. Wiley-Interscience, New York, 1987.

[2] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *Bell Sys. Tech. J.*, 61(8):1871–1894, 1982.

[3] J. Atlason, M. Epelman, and S. Henderson. Optimizing call center staffing using simulation and analytic center cutting plane methods. *Management Science*. To appear.

[4] F. Avram, D. Bertsimas, and M. Ricard. Fluid models of sequencing problems in open queueing networks; an optimal control approach. In F. P. Kelly and R. J. Williams, editors, *Stochastic Networks*, volume 71 of *The IMA volumes in mathematics and its applications*, pages 199–237, New York, 1995. Springer-Verlag.

[5] N. Bäuerle. Convex stochastic fluid programs with average cost. *J. Math. Anal. Appl.*, 259(1):137–156, 2001.

[6] N. Bäuerle. Discounted stochastic fluid programs. *Mathematics of Operations Research*, 26(2):401–420, 2001.

[7] Dimitris Bertsimas, David Gamarnik, and Jay Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: the holding cost objective. *Operations Research*, 51:798–813, 2003.

[8] Dimitris Bertsimas and Jay Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective. *Mathematical Programming*, 1:61–102, 2002.

[9] R. Billings. *A Heuristic Method for Scheduling and Dispatching of Factory Production Using Multiclass Fluid Networks.* PhD thesis, Graduate Program in Operations Research and Industrial Engineering, University of Texas at Austin, 2003.

[10] J.R. Birge and R.J.-B. Wets. Designing approximation schemes for stochastic optimization problems, in particular, for stochastic programs with recourse. *Mathematical Programming Study*, 27:54–102, 1986.

[11] H. Chen and D. Yao. *Fundamentals of Queueing Networks.* Springer, New York, 2001.

[12] J. G. Dai. Stability of fluid and stochastic processing networks. In *MaPhySto Miscellanea Publication, No. 9.* Centre for Mathematical Physics and Stochastics, 1999.

[13] J. G. Dai and G. Weiss. A fluid heuristic for minimizing makespan in job-shops. *Operations Research*, 50:692–707, 2002.

[14] J. Dupačová. Minimax stochastic programs with nonconvex nonseparable penalty functions. In A. Prékopa, editor, *Progress in Operations Research*, pages 303–316. Mathematica Societatis János Bolyai, Eger, Hungary, 1976.

[15] N.C.P. Edirisinghe and G-M. You. Second-order scenario approximation and refinement in optimization under uncertainty. *Annals of Operations Research*, 64:143–178, 1996.

[16] K. Frauendorfer. Solving SLP recourse problems with arbitrary multivariate distributions – the dependent case. *Mathematics of Operations Research*, 13:377–394, 1988.

[17] K. Frauendorfer and P. Kall. A solution method for slp recourse problems with arbitrary multivariate distributions—the independent case. *Problems of Control and Information Theory*, 17:177–205, 1988.

[18] H.I. Gassmann and W.T. Ziemba. A tight upper bound for the expectation of a convex function of a multivariate random variable. *Mathematical Programming Study*, 27:39–53, 1986.

[19] G. Gürkan. Simulation optimization of buffer allocations in production lines with unreliable machines. *Annals of Operations Research*, 93:177–216, 2000.

[20] G. Gürkan, F. Karaesmen, and Ö. Özdemir. Optimal threshold levels in stochastic fluid models via simulation based optimization. *Discrete*

Event Dynamical Systems., 2005. Submitted.

[21] J. M. Harrison and A. Zeevi. A method for staffing large call centers using stochastic fluid models. *Manufacturing & Service Operations Management.*, 7:20–36, 2005.

[22] C.C. Huang, W.T. Ziemba, and A. Ben-Tal. Bounds on the expectation of a convex function of a random variable: with applications to stochastic programming. *Operations Research*, 25:315–325, 1977.

[23] G. Iyengar and A. Zeevi. Parameter uncertainty implications on asymptotic analysis and design of stochastic systems. 2006. Preprint.

[24] J. R. Jackson. Networks of waiting lines. *Operations Research*, 5:518–521, 1957.

[25] P. Kall, A. Ruszczyński, and K. Frauendorfer. Approximation techniques in stochastic programming. In Y. Ermoliev and R.J.-B. Wets, editors, *Numerical Techniques for Stochastic Optimization*, pages 33–64. Springer-Verlag, Berlin, 1988.

[26] J. E. Kelley. The cutting plane method for solving convex programs. *SIAM Journal of Industrial and Applied Mathematics*, 8:703–712, 1960.

[27] Frank P. Kelly. Networks of queues with customers of different types. *J. Appl. Probab.*, 12:542–554, 1975.

[28] V. G. Kulkarni. Fluid models for single buffer systems. In J. H. Dshalalow, editor, *Frontiers in Queueing: Models and Applications in Science and Engineering*, pages 321–338, New York, 1997. CRC Press.

[29] A. Madansky. Bounds on the expectation of a convex function of a multivariate random variable. *Annals of Mathematical Statistics*, 30:743–746, 1959.

[30] V. I. Norkin, G. C. Pflug, and A. Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical Programming*, 63:425 –450, 1998.

[31] M. C. Pullan. An algorithm for a class of continuous linear programs. *SIAM J. Control and Optimization*, 31:1558–1577, 1993.

[32] M. C. Pullan. Forms of optimal solutions for separated continuous linear programs. *SIAM J. Control and Optimization*, 33:1952–1977, 1995.

[33] S. P. Sethi, H. Yan, H. Zhang, and Q. Zhang. Optimal and hierarchical controls in dynamic stochastic manufacturing systems: A survey. *Manufacturing & Service Operations Management*, 4(2):133–170, 2002.

[34] A. Shapiro. Monte Carlo sampling methods. In A. Ruszczyński and A. Shapiro, editors, *Stochastic Programming, Handbooks in Operations Research and Management Science.* Elsevier, Amsterdam, 2003.

[35] G. Sun, C. Cassandras, and C. Panayiotou. Perturbation analysis of multiclass stochastic fluid models. *Discrete Event Dynamical Systems*, 14:267–307, 2004.

[36] R.M. Van Slyke and R.J.-B. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[37] G. Weiss. On optimal draining of fluid reentrant lines. In Frank P. Kelly and Ruth J. Williams, editors, *Stochastic Networks*, volume 71 of *The IMA volumes in mathematics and its applications*, pages 93–105, New York, 1995. Springer-Verlag.

[38] Gideon Weiss. Optimal draining of fluid reentrant lines: some solved examples. In Frank P. Kelly, S. Zachary, and I. Zeidins, editors, *Stochastic Networks: Theory and Applications*, volume 4 of *Royal Statistical Society Lecture Note Series*, pages 19–34, Oxford, England, 1996. Oxford University Press.

[39] Ward Whitt. Staffing a call center with uncertain arrival rate and absenteeism. *Production and Operations Management*, 15(1):88–102, 2006.

Vita

Burak Buke was born in Corum, Turkey on 2 December 1980, the son of Mustafa and Dondu Buke. He received his Bachelor of Science degree in Industrial Engineering from Bogazici University, Istanbul, Turkey. During his undergraduate studies, he performed research on data mining and biometrics. In Fall 2002 he joined the Operations Research and Industrial Engineering program at the University of Texas at Austin as a graduate student. He received his Master of Science in Engineering degree from the same program in August 2007. During his doctoral study, he worked on analysis and optimization of systems with parameter uncertainty with Dr. John J. Hasenbein and Dr. David P. Morton. He also investigated revenue management problems arising in airlines industries. Burak joined the Industrial and Systems Engineering Department at the Ohio State University as a Lecturer in Fall 2007. He is currently teaching graduate level classes in stochastic processes and simulation at the OSU.

Permanent address: Bahcelievler M. Bahar 2. S.
No: 27/2 Corum/TURKEY

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.