GENERATING ROBUST AND STABLE MACHINE SCHEDULES FROM A PROACTIVE STANDPOINT

A DISSERTATION SUBMITTED TO

THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Selçuk GÖREN

August, 2009

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

_____

Prof. Dr. İhsan Sabuncuoğlu (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

_____

Prof. Dr. Selim Aktürk

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

_____

Prof. Dr. Meral Azizoğlu

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

_____

Prof. Dr. Erdal Erel

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

_____

Asst. Prof. Alper Şen

Approved for the Institute of Engineering and Science:

_____

Prof. Dr. Mehmet B. Baray

Director of the Institute

ABSTRACT

GENERATING ROBUST AND STABLE MACHINE SCHEDULES FROM A PROACTIVE STANDPOINT

Selçuk GÖREN
Ph.D. in Industrial Engineering
Supervisor: Prof. Dr. İhsan Sabuncuoğlu
August, 2009

In practice, scheduling systems are subject to considerable uncertainty in highly dynamic operating environments. The ability to cope with uncertainty in the scheduling process is becoming an increasingly important issue. In this thesis we take a proactive approach to generate robust and stable schedules for the environments with two sources of uncertainty: processing time variability and machine breakdowns. The information about the uncertainty is modeled using cumulative distribution functions and probability theory is utilized to derive inferences.

We first focus on the single machine environment. We define two robustness (expected total flow time and expected total tardiness) and three stability (the sum of the squared and absolute differences of the job completion times and the sum of the variances of the realized completion times) measures. We identify special cases for which the measures can be optimized without much difficulty. We develop a dominance rule and two lower bounds for one of the robustness measures, which are employed in a branch-and-bound algorithm to solve the problem exactly. We also propose a beam-search heuristic to solve large problems for all five measures. We provide extensive discussion of our numerical results.

Next, we study the problem of optimizing both robustness and stability simultaneously. We generate the set of all Pareto optimal points via ε-constraint method. We formulate the sub-problems required by the method and establish their computational complexity status. Two variants of the method that works with only a single type of sub-problem are also considered. A dominance rule and alternative

ways to enforce the rule to strengthen one of these versions are discussed. The performance of the proposed technique is evaluated with an experimental study. An approach to limit the total number of generated points while keeping their spread uniform is also proposed.

Finally, we consider the problem of generating stable schedules in a job shop environment with processing time variability and random machine breakdowns. The stability measure under consideration is the sum of the variances of the realized completion times. We show that the problem is not in the class $\mathcal{NP}$. Hence, a surrogate stability measure is developed to manage the problem. This version of the problem is proven to be $\mathcal{NP}$-hard even without machine breakdowns. Two branch-and-bound algorithms are developed for this case. A beam-search and a tabu-search based two heuristic algorithms are developed to handle realistic size problems with machine breakdowns. The results of extensive computational experiments are also provided.

Keywords: Single machine scheduling, job shop scheduling, robustness, stability, proactive scheduling, branch-and-bound, beam search, tabu search, ε-constraint method.

ÖZET

PROAKTİF BİR BAKIŞ AÇISINDAN GÜRBÜZ VE KARARLI MAKİNE ÇİZELGELERİ OLUŞTURULMASI

Selçuk GÖREN
Endüstri Mühendisliği, Doktora
Tez Yöneticisi: Prof. Dr. İhsan Sabuncuoğlu
Ağustos, 2009

Endüstride kullanılan çizelgeleme sistemleri işletme ortamlarındaki şartlar gereği ciddi miktarda değişkenlik ve belirsizlik etkisi altında çalışmaktadırlar. Çizelgeleme sırasında karşılaşılabilecek çeşitli belirsizliklerle baş edebilme niteliği günümüz koşullarında giderek önem kazanmaktadır. Bu tez çalışmasında proaktif bir yaklaşımla iki çeşit belirsizlik etkisindeki (işlem süresi değişkenliği ve makine arızalanması) ortamlarda gürbüz ve kararlı çizelgeler üretilmesi problemleri ele alınmaktadır. Belirsizlik hakkındaki bilgi, olasılık dağılımları aracılığıyla modellenmekte ve olasılık kuramı kullanılarak sistem hakkında çeşitli çıkarımlara ulaşılmaktadır.

İlk olarak tek makineli bir ortam ele alınmaktadır. İki gürbüzlük (beklenen toplam akış zamanı ve beklenen toplam gecikme) ve üç kararlılık (iş tamamlanma zamanları arasındaki farkların kareleri ve mutlak değerleri toplamı, iş tamamlanma zamanlarının toplam varyansı) ölçütü tanımlanmakta, bu ölçütlerin fazla zorlukla karşılaşmadan eniyilenebileceği özel durumlar tespit edilmektedir. Gürbüzlük ölçütlerinden biri için bir üstünlük kuralı, iki alt sınır ve problemi çözmek için bunları kullanan bir dal-sınır algoritması geliştirilmiştir. Her beş ölçüt için de büyük boyuttaki problemleri çözmek için kullanılabilecek bir demet taraması sezgiseli geliştirilmiş, kapsamlı sayısal deneylerle geliştirilen yöntemlerin performansları incelenmiştir.

Çalışılan ikinci problem tek makine ortamında kararlık ve gürbüzlüğün eşzamanlı eniyilenmesidir, Bütün Pareto optimum noktaları üreten bir ε-kısıt yöntemi

incelenmektir. Yöntemin gereksinim duyduğu alt problemler formüle edilmiş ve hesapsal karmaşıklıkları tespit edilmiştir. Yöntemin sadece tek bir cins alt probleme ihtiyaç duyan iki varyasyonu ele alınmış, bu varyasyonlardan birini güçlendirecek bir üstünlük kuralı ve bu kuralın değişik formülasyonları geliştirilmiştir. Önerilen tekniklerin performansları deneysel bir çalışmayla değerlendirilmiştir. Üretilen toplam nokta sayısını sınırlandırırken, noktaların dağılımını mümkün olduğunca eşit aralıklı tutacak bir yaklaşım da önerilmiştir.

Son olarak, işlem süresi değişkenliği ve rassal makine arızalanmalarına maruz atölye tipi işliklerde kararlı çizelgelerin oluşturulması problemi ele alınmaktadır. Kullanılan kararlılık ölçütü, iş tamamlanma zamanlarının varyansları toplamıdır. Bu problem $\mathcal{NP}$ sınıfında olmadığından vekil bir kararlılık ölçütü kullanılmıştır. Problemin bu halinin, makine arızalanmaları göz ardı edilse bile, $\mathcal{NP}$ –zor olduğu gösterilmiş ve tam çözüm yöntemi olarak iki dal-sınır algoritması geliştirilmiştir. Makine arızalanmalarını göz önüne alan ve büyük boyutlu örnekleri çözebilen biri demet taraması ve diğeri tabu araması olmak üzere iki sezgisel yöntem geliştirilmiştir. Geliştirilen yöntemler kapsamlı hesapsal deneylerle test edilmiştir.

Anahtar sözcükler: Tek makine çizelgeleme, atölye çizelgeleme, gürbüzlük, kararlılık, proaktif çizelgeleme, dal-sınır, demet taraması, tabu araması, ε-kısıt yöntemi

To my brother and sister through ties of love rather than ties of blood

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my advisor Prof. İhsan Sabuncuoğlu for his guidance, expertise, patience, tolerance, encouragement and unreserved support during my whole graduate study. His guidance and support was not invaluable only in academic issues but in all aspects of my graduate life. It has been an honor to work with him.

I am indebted to Prof. Selim Aktürk, Prof. Meral Azizoğlu, Prof. Erdal Erel and Asst. Prof. Alper Şen for accepting to be the members of my thesis committee, showing keen interest on the subject, devoting their valuable time to read and review this thesis and their valuable remarks and recommendations.

I am grateful to Dr. Hakan Gültekin for his friendship and academic and morale support.

I would like to express my deepest gratitude to all members of my family, especially to my mother Fatime Gören, for their love, understanding, patience and support.

Last but certainly not the least, my friends, buddies, brother and sister Utku Koç and Filiz Çınkır Koç deserve special mention for everything they brought to my life. Their patience, tolerance, encouragement, sympathy, confidence and academic and morale support were priceless and helped me get “unstuck” and move forward on many occasions. There is actually no way I can literally express my sincere gratitude to them, for words are inadequate. As an attempt, I have two things to say: I feel extraordinarily lucky for having them. I genuinely hope that they will remain being my everlasting brother and sister until the end of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scheduling is a decision-making process that is concerned with the allocation of limited resources (machines, material-handling equipment, operators, tools, fixtures, etc.) to competing tasks (operations of jobs) over time, with the goal of optimizing one or more objectives. The output of this decision process is time/machine/operation assignments. In the scheduling literature, the objective is generally to minimize functions such as makespan, tardiness, flow time, etc.

In practice, scheduling systems operate in dynamic and uncertain environments in which random interruptions prevent the execution of a schedule exactly as it is developed. Examples of such disruptions are machine breakdowns, rush orders, order cancellations, due-date changes, etc. Variability in processing times and other stochastic events further increase the variability in the system, which in turn deteriorate the scheduling performance.

Even though actual scheduling problems in real life are dynamic and stochastic, most of the existing literature addresses static and deterministic versions. But even these simplified problems (with deterministic and static assumptions) are $\mathcal{NP}$-hard or analytically intractable.

The uncertainties and dynamic nature of the real-world scheduling process can be seen as the major source of the gap between scheduling theory and practice. In the literature, several studies have been conducted to close this gap. In the early works, researchers employ a rolling horizon scheme to cope with the dynamic nature of scheduling environments, where the problem is successively solved using static

algorithms for different time windows (Nelson, Holloway, and Wong, 1977). The stochastic nature of scheduling has also been investigated in the literature. In these studies, uncertainty in job processing times, release times or due dates is modeled by probability distribution functions and formal probability theory is used to make inferences (Pinedo, 2002, Chapters 9-13). In the last two decades researchers have also proposed approaches including on-line scheduling, dynamic scheduling and real-time scheduling. Recently, two approaches to coping with uncertainty in the scheduling process have gained significant research interest: *reactive* and *proactive* scheduling. The objective in reactive scheduling is to revise schedules as unexpected events (disruptions) occur. On the other hand, proactive scheduling takes future disruptions into account while generating schedules.

The challenge of addressing the dynamic and stochastic nature of the scheduling process also affects the performance measure of choice. Although performance measures such as makespan, flow time, or tardiness have often been preferred in practice, in the recent literature two new measures are brought to the attention of practitioners: *robustness and stability*. These measures are particularly used in environments where uncertainty is a major issue.

Uncertainty has two kinds of major negative impacts on initial schedules. First, it degrades schedule performance. This effect is the topic of *robustness*. A schedule whose performance does not deteriorate in the face of disruptions is called *robust*. In other words, the performance of a robust schedule is expected to be insensitive to disruptions. In general, the performance of the realized schedule is the main concern of practitioners rather than the planned or estimated performance of the initial schedule. Hence, optimizing the former may be more appropriate than optimizing the latter and robustness is a practical performance measure. Second, unforeseen disruptions cause variability. This effect is the topic of *stability.* A schedule whose realization does not deviate from the original schedule in the face of disruptions is called *stable.* A schedule serves as a master plan for other shop-floor activities in addition to production, such as determining delivery dates, release times, and planning requirements for secondary resources such as tools, fixtures, etc. Any deviation from the production schedule can disrupt these secondary activities and increase system nervousness. Thus, stability is also an important measure in practice.

Robustness and stability can be illustrated with the help of Figure 1.1. The top Gantt chart in the figure depicts a possible initial schedule for a job-shop environment with three jobs and three machines subject to random breakdowns. The bottom Gantt chart shows a possible realization of the initial schedule. The shaded area on the realized schedule of machine 2 between times 7 and 9 represents a breakdown. Assume that the performance measure of interest is the maximum completion time (C_{max}). From the robustness viewpoint, the scheduler should be concerned with the performance of the realized schedule (C_{max} = 13 in the example) rather than the performance of the initial schedule (C_{max} = 10 in the example). Hence, he/she optimizes a measure (*robustness measure*) that is defined on the *realized* schedule. Another way to look at this is to minimize the performance deviation between the initial and the realized schedules (ΔC_{max} = 13 – 10 = 3 in the example). Observe that the operation of job 1 on machine 2 completes later than planned. Similarly, while the operation of job 2 on that machine is planned to be processed between times 8 and 10, it is actually processed between times 11 and 13 because of the breakdown. From the stability viewpoint, such deviations from the initial schedule (i.e., the master plan) should be minimized. Hence, the scheduler optimizes a measure (*stability measure*) defined in terms of the deviations between the initial and the realized schedules.

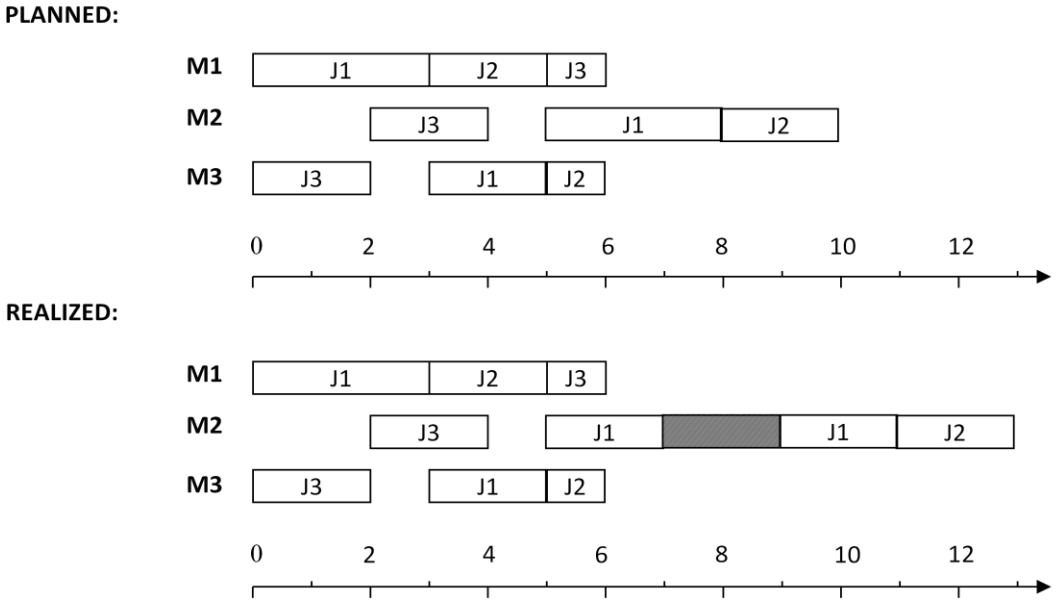


Figure 1.1. An Initial Schedule and its Realization for J3| |Cmax

The reactive and proactive scheduling approaches and these two new performance measures (robustness and stability) are discussed in more detail in (Sabuncuoglu and Goren, 2009).

In this thesis, we study generating robust and stable schedules in the face of processing time variability and random machine breakdowns. To form the basis for the later parts, we start with a comprehensive review of the relevant literature in Chapter 2.

We first consider the single machine environment in Chapter 3. In case of a breakdown, the machine is unavailable until it is repaired. The times for repair are also random and independent of each other and of the breakdown process. A job preempted due to a breakdown is processed for its remaining processing time (i.e., preempt-resume policy is assumed). No other preemptions are allowed. We take a proactive point of view and define several robustness and stability measures.

Although there are some studies which measure robustness as a minimax regret (e.g., Daniels and Kouvelis, 1995), the majority of recent studies on robustness involve expected realized performance. The expected realized performance can be the robustness measure by itself (e.g., Wu *et al.*, 1999) or can be a part of it (e.g., Leon *et al.*, 1994). In this thesis, we use the expected realized performance measure as the robustness measure. We consider two performance measures: expected total flow time (*RM1*) and expected total tardiness (*RM2*).

The most frequent way to measure the deviation between the initial and the realized schedules (stability) is to compare their job completion times (Wu *et al.* 1993, Mehta and Uzsoy, 1998). We use two stability measures based on this comparison: the sum of the squared differences (*SM1*) and the sum of the absolute differences (*SM3*). We also use the sum of the variances of the realized completion times as another stability measure (*SM2*).

We also derive optimality conditions and propose a proactive branch-and-bound (B&B) algorithm, which uses a stochastic dominance rule, for minimizing the expected total tardiness (*RM2*). We first consider a single machine environment because of its simplicity and the possible extendibility of its results to more realistic multi-machine environments.

In Chapter 4, we focus on optimizing robustness and stability simultaneously in a single machine environment with random processing times. There are two general approaches to multicriteria optimization. One is to combine the individual criteria into a single composite criterion. The other is to generate a set of solutions that contains an optimal solution for each reasonable composite criterion that one can think of (the set of Pareto optimal solutions). Evans (1984) identifies these approaches as *a priori* and *a posteriori* optimization, respectively. In Chapter 4, we consider a posteriori optimization of robustness and stability simultaneously. We generate the set of all Pareto optimal points via so called ε-constraint method. We formulate the sub-problems required by the method and establish their computational complexity status. Two variants of the method that works with only a single type of sub-problem are also considered. A dominance rule and alternative ways to enforce the rule to strengthen one of the single sub-problem versions of the method are discussed. The performances of the methods and the dominance rules are evaluated in an experimental study.

In Chapter 5, we consider the job shop environment. Unlike the previous studies in the literature and the preceding chapters, stability is taken to be the only (thus primary) objective function to be optimized. Operation processing times as well as machine up and down times are taken as random variables. We use the sum of the variances of the realized completion times as the stability measure (*SM*). We call the problem of minimizing *SM* in a job shop environment subject to random machine breakdowns and processing time variability as the problem Π. As shown in Chapter 5, Π is not in the class $\mathcal{NP}$. Hence, a surrogate stability measure (*SSM*) is developed to manage the problem. This version of the problem is called Π′. The problem of minimizing *SSM* in a job shop environment subject to processing time variability only (i.e., no machine breakdowns) is called the problem Π″. It is proven that Π″ (and therefore Π′) is $\mathcal{NP}$-hard. Two exact solution procedures (branch-and-bound algorithms) are developed for Π″. Two heuristics (a beam-search and a tabu-search algorithm) are also developed to handle large instances of Π″. As will be shown, calculation of even the surrogate measure (*SSM*) is not possible for Π′ due to random machine breakdowns. Thus, the beam-search and tabu-search algorithms are modified to handle breakdowns case. The same modifications cannot be applied to the branch-and-bound algorithms due to the following two reasons: first, they would lose the property of being exact solution procedures and they are computationally too

expensive to use as heuristics, and second, the proposed tabu-search algorithm already performs significantly well, even better than the branch-and-bound algorithms.

Chapter 5 extends the stability scheduling literature in four ways: first, a new practical stability measure is defined; second, complexity status of the problems are determined; third, processing time variability and machine breakdowns are simultaneously considered in the problem settings; and finally, two exact solution procedures and heuristics are proposed to solve the problems.

Finally we outline our contributions and discuss some future research directions in Chapter 6.

Chapter 2

Literature Review

Although this thesis is on schedule robustness and stability, the literature on scheduling with unreliable machines is relevant. We first review a few studies in Section 2.1. In Section 2.2 we review some studies in reactive scheduling literature to give the reader a flavor of this line of research. Subsequently, we review the studies in the literature that explicitly address robustness or stability of schedules in Section 2.3. Finally, we briefly discuss how this thesis contributes to the available literature in Section 2.4.

2.1 Scheduling with Machine Availability Constraints

Adiri *et al.* (1989) consider the problem of minimizing total flow time in a single machine environment subject to random breakdowns. In contrast with our study, only one machine breakdown occurs and a preempt-repeat policy is assumed. The authors show that if the distribution function of the time to breakdown is concave, then the shortest processing time first (SPT) rule stochastically minimizes the flow time. For the case of multiple breakdowns, it is proven that SPT minimizes the expected flow time when the times to breakdown are exponentially distributed. The authors show that the problem is $\mathcal{NP}$-hard when the time for the single breakdown is known in advance and the processing times of the jobs are deterministic.

In a later study, Adiri *et al.* (1991) consider the single machine scheduling problem with deterministic processing times and due dates subject to a single random breakdown. The authors develop policies to minimize the number of tardy jobs stochastically, working under certain assumptions for both preempt-resume and preempt-repeat policies.

Similar to our study in Chapter 3, Li and Glazebrook (1998) consider the single machine scheduling problem with random processing times and multiple machine breakdowns with a preempt-resume policy. The objective is to minimize a weighted sum of an increasing function of the completion times in expectation. The authors develop a dominance rule based on pairwise interchanges of adjacent jobs. The rule is also relaxed to allow uptimes to be distributed as a mixture of exponentials and according to a Gamma distribution. The dominance rule, however, cannot be applied to due-date related measures, which are not functions of completion times only. We develop a similar dominance rule based on pairwise interchanges of jobs (not necessarily adjacent) for the total tardiness measure in case of no machine breakdowns in Chapter 3.

Li *et al.* (1998) consider the same problem under Erlang uptime distribution. All jobs are assumed to have a common exponentially distributed due date (compared with deterministic but different due dates in our study). The authors develop dominance rules based on pairwise interchanges of adjacent jobs in order to minimize the weighted number of tardy jobs, weighted flow times, and weighted sum of job delays.

Leung and Pinedo (2004) study the preemptive parallel machine scheduling problem with random breakdowns and deterministic processing times and due dates. The authors develop conditions on the number of available machines $m(t)$ that minimize total completion time, makespan, or maximum lateness. The authors also analyze cases with deadlines and precedence constraints.

We refer interested readers to Pinedo (2002) to see a concise summary of stochastic scheduling results. Next, we review the studies in the literature that explicitly address robustness or stability of schedules.

2.2 Reactive Scheduling

In reactive scheduling literature, several other authors develop schedules in the face of disruptions without considering disruption in the decision making phase. Here we review some of the recent studies to give the reader a flavor of this line of research. The interested reader is referred to Sabuncuoglu and Bayiz (2000), Vieira, Herrmann, and Lin (2003) and Aytug *et. al.* (2005) for a broader literature review.

Church and Uzsoy (1992) analyze the performance of *event-driven scheduling* in a single machine environment with dynamic job arrivals. They classify the events that change the system state into two categories: 1) the events that require immediate response (exceptions) and 2) the events that can be ignored until the next rescheduling point. The schedule is revised periodically but scheduling is also triggered when an exception occurs. In their model, the exceptions are arrivals of jobs with tight due dates. For each job i arriving between times $(k-1)T$ and T, they calculate the slack $s_i = d_i - r_i$, where d_i is the due-date r_i is the ready time of job i and T is the period length. If this value is smaller than a constant w (window length), the arrival of job i is considered as an exception and a scheduling decision is triggered. A schedule is also generated at the beginning of each period (at the times kT). The authors use EDD dispatching rule to generate schedules at each revision. The performance measure is maximum lateness. Their computational experiments indicated that benefits of extra scheduling diminish rapidly. They conclude a well-designed event-driven policy can achieve good system performances with less computational burden as compared to scheduling in response to every event that change the system state.

Akturk and Gorgulu (1999) study on the rescheduling of operations in a modified flow shop environment in response to a machine breakdown. In a modified flow shop, jobs can enter the system at one of the several machines, can progress through the system by a limited number of paths and can exit the system on one of the several machines. Hence, it falls somewhere between a flow shop and a job shop. The authors assume that an initial schedule is available and it is followed until a single machine breakdown occurs. In response to the machine breakdown, they reschedule the operations to match up with the initial schedule at a point in the future. In the first stage, they determine a match-up point for each machine. Then the authors decompose

the rescheduling problem into three parts: 1) the scheduling of the down machine, 2) the scheduling of the machines in the upward direction of the down machine and 3) the scheduling of the machines in the downward direction of the down machine. If a resulting schedule is not feasible, then the match-up point is changed to enlarge the set of jobs that are rescheduled. Their experimental results indicate that the proposed algorithm is very effective in terms of schedule efficiency, computational times and schedule stability.

In another study, Sabuncuoglu and Karabuk (1999) investigate the scheduling/rescheduling problem in an flexible manufacturing system (FMS) environment. The authors propose a filtered beam search. For several reactive scheduling policies in response to machine breakdowns and processing time variability, the authors compare off-line and on-line scheduling algorithms. Their computational experiments indicate that the proposed off-line algorithm performs better than on-line machine and several AGV scheduling rules, under all experimental conditions for the makespan, mean flow time and mean tardiness criteria. They also show that it is not always beneficial to reschedule the operations in response to every unexpected event. They conclude that the periodic response with an appropriate period length can be effective to cope with the interruptions.

Sabuncuoglu and Bayiz (2000) study the reactive scheduling problem in a job shop environment. The authors measure the effect of shop floor configuration (system size and load allocation) on the performance of the scheduling methods (off-line and on-line). Their performance criteria are makespan and mean tardiness. In the first part of the study, they compare a beam search based heuristic to other well-known algorithms. In the second part, they study on different reactive policies such as partial scheduling versus full scheduling, etc. Their computational experiments indicated that beam search is quite promising for the job shop problem and partial offline scheduling can be a very practical tool in a highly dynamic and stochastic environment.

In the next section we review the studies that explicitly address robustness and stability.

2.3 Schedule Robustness and Stability

The studies on schedule robustness and stability can be divided into two parts - those that model uncertainty by probability density functions, and those that hedge against the worst contingency that may arise without considering any specific probability distribution. The latter is known as the *robustness approach* in the literature. In both approaches, the source of uncertainty is either the variability of task processing times or machine availability (the machines are subject to a breakdown/repair process).

Leon *et al.*'s 1994 study is an example of the first approach. They consider the job-shop scheduling problem with machine breakdowns. The objective is to construct a robust initial schedule. The robustness measure for a schedule is calculated as a convex combination of the expected makespan of the realized schedule and the expected deviation from the initial deterministic makespan. In a job shop environment with multiple machine failures, however, calculating this measure analytically is intractable. They develop a surrogate measure and minimize that measure instead. The results indicate that the proposed algorithm outperforms the classical algorithms that focus on minimizing makespan only.

Wu *et al.* (1999) propose a graph-theoretic decomposition for the job shop scheduling problem to achieve schedule robustness. Expected average weighted tardiness is used as the robustness measure. The authors use a graph representation of this problem, in which conjunctive arcs represent precedence constraints and disjunctive arcs join operations competing for the same resource. They propose a branch-and-bound algorithm that processes disjunctive arcs and changes *some* of them into conjunctive arcs. This effectively fixes some of the scheduling decisions. The remaining scheduling decisions are made dynamically by applying the apparent tardiness cost (ATC) heuristic (Vepsalainen and Morton, 1987). Their computational experiments indicate that this scheme displays better robustness performance under a wide range of disturbance levels (various levels of processing time variability) compared to traditional off-line and on-line methods.

There are also studies that model uncertainty with probability density functions with the aim of generating stable schedules. For example, Wu *et al.* (1993) study the

single machine rescheduling problem under machine disruptions. They reschedule the jobs in response to each machine failure so that a minimum makespan is obtained with high schedule stability (the measure they use is similar to *SM3* in Chapter 3). Since the problem is $\mathcal{NP}$-hard even without stability considerations, they use a pairwise swapping heuristic and a genetic algorithm to generate a list of non-dominated schedules. Their computational results show that the stability of the schedules could be improved significantly with little sacrifice in makespan.

Mehta and Uzsoy (1998, 1999) generate initial stable schedules under random machine breakdowns. Their objective is to generate an initial schedule with minimal deviation (i.e., *SM3*) while keeping shop floor performance degradation at an acceptable level. The specific problem they study in the first paper is the single machine scheduling problem where jobs have unequal ready times and random machine breakdowns are present. In the second paper, they study the job shop scheduling problem with random machine breakdowns. In both studies, they use maximum lateness as the shop floor performance measure. Unlike Wu *et al.* (1993), they consider the minimization of the deviation between the initial and the realized schedule while generating an initial schedule, not when rescheduling after a breakdown. The authors offer a two-stage approach. In the first stage, a job sequence that will minimize the maximum lateness is determined. In the second stage, they insert idle times into the sequence. Their computational results indicate that stability can be easily improved while slightly increasing maximum lateness.

O'Donovan *et al.* (1999) combine the reactive and the proactive approaches and examine the scheduling/rescheduling policy using stability and efficiency measures in a single machine environment. Schedule efficiency is measured by total tardiness (*RM2* in Chapter 3). Stability is measured by absolute completion time deviations from the initial schedule (*SM3* in Chapter 3). The system under study has non-zero job ready times and random machine breakdowns. This study is similar to the one by Mehta and Uzsoy (1999) except that total tardiness is used instead of maximum lateness. They consider pure ATC and ATC with inserted idle times for initial schedule generation. Rescheduling alternatives are ATC, a modified ATC (which calculates the slack of a job based on its predicted completion time, taking inserted idle times into account) and right-shift scheduling. Their results indicate that ATC

with inserted idle times for an initial schedule and the modified ATC for rescheduling are the best for stability.

For the robustness approach, we refer the reader to Kouvelis and Yu (1996), who apply this method to various problems such as linear programming, assignment problem, shortest path problem, etc. as well as scheduling. An example of such an approach in the machine scheduling context is the study of Daniels and Kouvelis (1995). They generate initial robust schedules to hedge against processing time variability in a single machine environment. The authors propose a scenario-based representation and analysis of uncertainty rather than using stochastic models. They use a policy that finds the schedule whose performance degradation in its worst-case scenario is the least among all feasible schedules (i.e., minimax regret strategy in decision theory). The authors study a single machine problem where the performance measure is total flow time, and the source of uncertainty is processing time variability. The authors prove that a properly selected finite set of scenarios is enough to determine the worst-case absolute deviation of a given sequence and construct a procedure that calculates the worst-case evaluation in polynomial time. They develop a branch-and-bound algorithm and two $O(n \log n)$ surrogate relaxation heuristics that utilize this procedure to generate robust schedules. The authors compare their solutions to the SEPT (shortest expected processing time) solution, which is used in practice to generate an optimal sequence of jobs. They observe that SEPT performs poorly in terms of robustness.

Such a minimax regret approach to robustness may be more appropriate than the more frequently used expected performance measure approach if the distributions that capture the uncertainty are unknown or imprecise. Additionally, in many cases a stochastic approach that models the uncertainty with probability density functions assumes distributional independence to improve analytical tractability. If such an assumption is invalid (i.e., strong correlations exist among the probability distributions), a minimax regret approach may be more suitable to employ. Finally, if the scheduling decisions are evaluated *ex post* (as if all the relevant information had been known in advance of scheduling), a decision maker may be inclined to reduce the difference between the realized performance and the optimal performance that could have been achieved (i.e., minimize regret), rather than the average performance (Daniels and Kouvelis, 1995).

Sotskov *et al.* (1997) introduce another viewpoint for stability. They handle the uncertainty in a job shop environment by an *a posteriori* analysis, in which an optimal schedule has already been constructed and the challenge is to determine the maximum variation in the processing time of the operations such that the optimal schedule at hand still remains optimal. Such a maximum variation is called *the stability radius* of the schedule. This notion of stability, obtained by sensitivity analysis, can be considered as a measure of *solution robustness* as per of Herroelen and Leus (2005). Although this type of post-optimality analysis may provide some valuable insights about the impacts of uncertainty, it is also associated with some problems. If the stability radius of the optimal schedule is large enough to accommodate all possible changes in the processing times, the optimal schedule at hand can safely be used, but if it is not that large, the question of what course of action to take remains to be answered. Hence, in this thesis, we take a proactive stance and incorporate uncertainty into the scheduling processes. We concentrate on optimizing the *quality robustness* rather than the solution robustness.

2.4 Discussion

In this thesis we optimize explicitly defined robustness and stability measures in a proactive fashion. In general, calculating actual robustness and stability measures analytically is very difficult. For that reason, in the previous studies researchers employ surrogate measures to indirectly calculate the robustness or stability of a schedule. The surrogate measures used in the existing studies, however, are not sophisticated enough to incorporate the known information about the uncertainty adequately, as also stated in Mehta and Uzsoy (1998). In this thesis, we use the probability theory to derive inferences about minimizing robustness or stability measures and try to fill this gap.

Specifically, in Chapter 3, we solve the problem for a number of special cases in the single machine environment. For intractable cases, instead of employing surrogate measures, we use a beam-search (BS) algorithm developed in this chapter that employs simulation to calculate robustness or stability measures. Thus, we use the

available information about the uncertainty better than does the indirect approach of employing surrogate measures. Moreover, in the previous studies, makespan or maximum lateness is used as the performance measure for the sake of simplicity. In Chapter 3, however, we consider flow time and tardiness criteria, as they are used more often in practice.

Even though scheduling with more than one objective has been studied since 1980s, optimizing robustness and stability simultaneously in a proactive way is not thoroughly considered in the literature. The previous studies either preferred including stability into the picture later in the reactive phase after an initial schedule is at hand (e.g., Wu *et al.*, 1993) or stability alone is optimized by inserting additional idle time into the schedules with the hope that the primary objective does not worsen a lot (e.g., Mehta and Uzsoy, 1998, 1999). In Chapter 4, we consider both measures at the same time proactively.

Generally speaking, the stability literature is rather thin and the only source of the uncertainty that is considered is the presence of machine breakdowns. In Chapter 5, a new and practical stability measure is considered in a job shop scheduling environment subject to random machine breakdowns and processing time variability. Exact solution procedures and heuristics are provided.

We can now conclude the review of the existing literature and continue with our contributions in the rest of the thesis.

Chapter 3

Single Machine Environment

3.1 Introduction

In this chapter, we take a proactive scheduling approach to study the single machine scheduling problem with two sources of uncertainty: processing time variability and machine breakdowns. The reason for starting with a single machine environment is that it is a special case of all other environments. The results that can be obtained in this simple environment can provide insights and can form a basis for more complicated multi-machine environments.

We define several robustness and stability measures in this chapter. As reviewed in Chapter 2, two kinds of robustness measures have been used in the literature: based on regret and based on realized performance. In this chapter, we use the expected realized performance measure as the robustness measure. We consider two performance measures: expected total flow time (*RM1*) and expected total tardiness (*RM2*).

The most frequent approach to measure the deviation between the initial and the realized schedules (stability) is to compare their job completion times (Wu *et al.* 1993, Mehta and Uzsoy, 1998). We use two stability measures based on this comparison: the sum of the squared differences (*SM1*) and the sum of the absolute differences (*SM3*). We also use the sum of the variances of the realized completion times as another stability measure (*SM2*). The rationale behind this and how it corresponds to the difference between the initial and the realized schedules are explained in Section

3.5.1. Note that all these stability measures can be trivially minimized by inserting large blocks of idle times between jobs in the initial schedule. In this chapter, however, we confine ourselves to the class of non-delay schedules as inserting idleness deteriorates robustness performance.

We also derive optimality conditions and propose a proactive branch-and-bound (B&B) algorithm, which uses a stochastic dominance rule, for minimizing the expected total tardiness (*RM2*). We consider a single machine environment because of its simplicity and the possible extendibility of its results to more realistic multi-machine environments.

The rest of this chapter is organized as follows. In Section 3.2 and 3.3, we approach the proactive scheduling problem in a single machine environment using probability theory. The robustness and stability measures are discussed in Sections 3.4 and 3.5, respectively. In Section 3.6, we present a branch-and-bound algorithm that utilizes insights gained in the previous analysis to minimize the expected total tardiness in a single machine environment with variable processing times. We present a bream search algorithm that can handle other performance measures and machine breakdowns in Section 3.7. Section 3.8 is dedicated to the assessment of the performance of the proposed algorithms with computational experiments. Finally, we make concluding remarks and discuss future research directions in Section 3.9.

3.2 Notation

We consider the single machine scheduling problem with random processing times and machine breakdowns. The uptimes have independent and identical general distribution $G_1(t)$. Similarly, the down times (i.e., the times that the machine is not in operation due to breakdown) are independent and identically distributed according to a general distribution $G_2(t)$. The processing times of the jobs are all random variables with known general distribution functions that may differ from job to job. Let $H_j(t)$ be the processing time distribution of job j. Let the random variable C_j denote the completion time of job j in the realized schedule. Let X_j denote the processing time of job j. We assume that all n jobs are released at time $t = 0$. Let U_1, U_2 … be the

sequence of uptimes and D_1, D_2… be the sequence of downtimes. That is, the machine is operational from time 0 until U_1, when the first breakdown occurs. The machine then takes time D_1 to be repaired and is again available for processing from time $U_1 + D_1$ until time $U_1 + D_1 + U_2$, and so on. We denote this stochastic single machine scheduling problem as $1 \mid X_j \sim H_j(t)$; *brkdwn:* $U \sim G_1(t)$, $D \sim G_2(t)$; $\beta \mid \gamma$, where $1 \mid \beta \mid \gamma$ denotes the deterministic version. Here, β is the set of scheduling attributes, such as release dates, presence of sequence dependent setup times, preemptions, precedence constraints, etc. and γ is the objective function. If breakdowns were not present, the notation would be $1 \mid X_j \sim H_j(t)$; $\beta \mid \gamma$.

Define $N(t) = \sup\{k \geq 0 \mid \sum_{i=0}^{k} U_i \leq t\}$, where $U_0 := 0$. That is, $N(t)$ is the number of machine breakdowns that occur up to total busy time t. Note that $N(t)$ is increasing in t. Here, we consider the case where the machine can be down more than once during the processing of a job and the job is processed for its remaining processing time after each breakdown (i.e., the work done on a job is not lost).

Y_j denotes the time that job j occupies the machine, including the processing time of the job and all the repair times during which the job stays on the machine. Let R_{jk} denote the k^{th} repair time during the processing of job j. Since R_{jk}'s are i.i.d., let $r = E[R_{jk}] = \int_0^\infty t dG_2(t)$ and $v = Var[R_{jk}] = \int_0^\infty (t-r)^2 dG_2(t)$. Let B_j denote the number of machine failures during the processing of job j. Then, we have $Y_j = X_j + \sum_{k=1}^{B_j} R_{jk}$.

We first begin by a definition and several propositions, which will be used in the treatment of the robustness and stability measures in Sections 3.4 and 3.5, respectively.

3.3 Preliminaries

Definition 3.1 (Ross, 1983). *A random variable V is said to be stochastically larger than a random variable W, written* $V \geq_{st} W$, *if* $P\{V > a\} \geq P\{W > a\}$ *for all a.*

Proposition 3.1. *Let $V_1,\ldots,V_n$ be independent and $W_1, \ldots, W_n$ be independent. If $V_i \geq_{st} W_i$ for all i, then for any increasing f, $f(V_1,\ldots,V_n) \geq_{st} f(W_1,\ldots,W_n)$.*

Proposition 3.2. *If $V \geq_{st} W$ then* $\max\{V, 0\} \geq_{st} \max\{W, 0\}$.

We refer the reader to Example 8.2(a) and Question 8.1 in Ross (1983) for the proofs of these two propositions. Both proofs involve the *coupling method*, which is explained in Ross's Chapter 8.

Proposition 3.3. *If up times are exponentially distributed with the rate λ, then*

$$\begin{aligned} E[B_j] &= \lambda E[X_j] \\ E[B_j^2] &= \lambda E[X_j] + \lambda^2 E[X_j^2] \\ Var[B_j] &= \lambda E[X_j] + \lambda^2 Var[X_j] \end{aligned}$$

Proof.

$$\begin{aligned} E[B_j] &= \int_0^\infty E[B_j \mid X_j = t] dH_j(t) \\ &= \int_0^\infty \lambda t dH_j(t) \\ &= \lambda E[X_j] \end{aligned}$$

$$\begin{aligned} E[B_j^2] &= \int_0^\infty E[B_j^2 \mid X_j = t] dH_j(t) \\ &= \int_0^\infty (\lambda t + \lambda^2 t^2) dH_j(t) \\ &= \lambda E[X_j] + \lambda^2 E[X_j^2] \end{aligned}$$

$$\begin{aligned} Var[B_j] &= E[B_j^2] - (E[B_j])^2 \\ &= \lambda E[X_j] + \lambda^2 E[X_j^2] - \lambda^2 (E[X_j])^2 \\ &= \lambda E[X_j] + \lambda^2 Var[X_j] \end{aligned}$$

□

Proposition 3.4. *If up times are exponentially distributed with the rate λ, then*

$$\begin{aligned} E[Y_j] &= (1+\lambda r)E[X_j] \\ Var[Y_j] &= \lambda(v+r^2)E[X_j]+(1+\lambda^2 r^2)Var[X_j] \end{aligned}$$

Proof.

$$\begin{aligned} E[Y_j] &= E[X_j]+E[\sum_{k=1}^{B_j} R_{jk}] \\ &= E[X_j]+E[B_j]E[R_{jk}] \\ &= E[X_j]+\lambda E[X_j]r \\ &= (1+\lambda r)E[X_j] \end{aligned}$$

$$\begin{aligned} Var[Y_j] &= Var[X_j]+Var[\sum_{k=1}^{B_j} R_{jk}]+2Cov(X_j,\sum_{k=1}^{B_j} R_{jk}) \\ &= Var[X_j]+Var[\sum_{k=1}^{B_j} R_{jk}]+2(E[X_j\sum_{k=1}^{B_j} R_{jk}]-E[X_j]E[\sum_{k=1}^{B_j} R_{jk}]) \\ &= Var[X_j]+Var[\sum_{k=1}^{B_j} R_{jk}]+2E[X_j\sum_{k=1}^{B_j} R_{jk}]-2\lambda r(E[X_j])^2 \\ &= Var[X_j]+E[B_j]Var[R_{jk}]+(E[R_{jk}])^2 Var[B_j] \\ &\quad +2E[X_j\sum_{k=1}^{B_j} R_{jk}]-2\lambda r(E[X_j])^2 \\ &= Var[X_j]+\lambda E[X_j]Var[R]+r^2(\lambda E[X_j]+\lambda^2 Var[X_j]) \\ &\quad +2E[X_j\sum_{k=1}^{B_j} R_{jk}]-2\lambda r(E[X_j])^2 \\ &= \lambda(Var[R]+r^2)E[X_j]+(1+\lambda^2 r^2)Var[X_j] \\ &\quad -2\lambda r(E[X_j])^2+2E[X_j\sum_{k=1}^{B_j} R_{jk}] \end{aligned} \tag{3.1}$$

Since we have

$$\begin{aligned}
E[X_j \sum_{k=1}^{B_j} R_{jk}] &= \sum_{m=0}^{\infty} E[X_j \sum_{k=1}^{B_j} R_{jk} \mid B_j = m] P\{B_j = m\} \\
&= \sum_{m=0}^{\infty} E[X_j \sum_{k=1}^{m} R_{jk}] \int_0^{\infty} P\{B_j = m \mid X_j = t\} dH_j(t) \\
&= \sum_{m=0}^{\infty} mrE[X_j] \int_0^{\infty} \frac{e^{-\lambda t}(\lambda t)^m}{m!} dH_j(t) \\
&= rE[X_j] \int_0^{\infty} \sum_{m=0}^{\infty} m \frac{e^{-\lambda t}(\lambda t)^m}{m!} dH_j(t) \\
&= rE[X_j] \int_0^{\infty} \lambda t dH_j(t) \\
&= \lambda r (E[X_j])^2,
\end{aligned}$$

the last two terms in (3.1) cancel out and we have

$$Var[Y_j] = \lambda(Var[R] + r^2)E[X_j] + (1 + \lambda^2 r^2)Var[X_j] \qquad \square$$

3.4 Robustness

As mentioned in the literature review, most frequently used approaches for measuring robustness involve expected realized performance in one way or another. Similar to the previous studies in the literature, in this chapter the robustness of schedules is assessed in terms of expected performance measures. We consider two performance measures: expected total flow time (*RM1*) and expected total tardiness (*RM2*). We begin with the flow time case.

3.4.1 Total Flow Time

Recall that *RM1* is the expected realized total flow-time. That is, $RM1 = E[\sum_{j=1}^{n} C_j]$.

Minimizing expected total weighted flow time in a single machine environment subject to random machine breakdowns is known to be $\mathcal{NP}$-hard (Adiri *et al.*, 1989). Even though the status of the unweighted case is unknown, it can be said that the

problem is analytically intractable, for it is difficult even to calculate the objective function value of a given solution. We present an optimality condition that holds in a special case here.

Theorem 3.1. *If $X_j \leq_{st} X_{j+1}$ for $j = 1,\ldots,n-1$, the job sequence $\{1,\ldots,n\}$ i.e., SSPT (stochastically smallest processing time) order is an optimal solution to $1 \mid X_j \sim H_j(t)$; brkdwn: $U \sim G_1(t)$, $D \sim G_2(t) \mid RM1$ problem.*

Proof. Consider an optimal sequence S. Assume that there exists a pair of adjacent jobs i and j such that $X_j \leq_{st} X_i$ and job j succeeds job i in S. Because if such a pair does not exist, either S is already the sequence $\{1,\ldots,n\}$ or it can be put into that form by simply swapping the labels of the jobs whose processing times have the same distribution. Therefore, without loss of generality we assume that there exists such a pair. Now consider a sequence S', obtained from S by swapping the positions of jobs i and j. We compare $RM1(S)$ and $RM1(S')$. We may ignore the jobs other than i and j in this comparison, since nothing changes for them. Let their contribution to the objective function be c. Let T denote the sum of the processing times of the jobs that precede i in S. We have

$$RM1(S) = E[T + X_i + \sum_{k=1}^{N(T+X_i)} D_k + T + X_i + X_j + \sum_{k=1}^{N(T+X_i+X_j)} D_k] + c$$

and

$$RM1(S') = E[T + X_j + \sum_{k=1}^{N(T+X_j)} D_k + T + X_i + X_j + \sum_{k=1}^{N(T+X_i+X_j)} D_k] + c\,.$$

Hence,

$$RM1(S) - RM1(S') = E[X_i - X_j] + E[\sum_{k=1}^{N(T+X_i)} D_k - \sum_{k=1}^{N(T+X_j)} D_k]\,.$$

Since $X_j \leq_{st} X_i$ and $N(t)$ is increasing, $N(T + X_j) \leq_{st} N(T + X_i)$ by Proposition 3.1. By coupling we also have $\sum_{k=1}^{N(T+X_j)} D_k \leq_{st} \sum_{k=1}^{N(T+X_i)} D_k$, and therefore $RM1(S) - RM1(S') \geq 0$. This means that S' is also an optimal solution. If we continue interchanging positions of adjacent jobs in this manner until no pair of adjacent jobs i and j such that $X_j \leq_{st} X_i$ and job j succeeds job i exists, we obtain a series of optimal solutions. The last solution we obtain is either already the sequence $\{1,\ldots,n\}$ or it can be put into that

form by simply swapping the labels of the jobs whose processing times have the same distribution. □

Corollary 3.1. *SEPT (Shortest Expected Processing Time) order gives an optimal solution for 1 | X_j ~ exponential(λ_j); brkdwn: U ~ $G_1(t)$, D ~ $G_2(t)$ | RM1.*

Corollary 3.2. *SEPT (shortest expected processing time) order gives an optimal solution for 1 | X_j ~ $H_j(t)$ | RM1.*

Corollary 3.1 and 3.2 are known results in the literature. See Pinedo (2002, Chapter 10). Theorem 3.1 can also be deducted from the dominance rule developed by Li and Glazebrook (1998).

3.4.2 Total Tardiness

RM2 is the expected realized total tardiness. That is, $RM2 = E[\sum_{j=1}^{n} \max(0, C_j - d_j)]$, where d_j is the due date of job *j*.

Theorem 3.2. *1 | X_j ~ $H_j(t)$; brkdwn: U ~ $G_1(t)$, D ~ $G_2(t)$ | RM2 is $\mathcal{NP}$-hard.*

Proof. We reduce $1||\sum_j T_j$ to *1 | X_j ~ $H_j(t)$; brkdwn: U ~ $G_1(t)$, D ~ $G_2(t)$ | RM2*. Begin with a $1||\sum_j T_j$ instance. Take all repair times as zero. Do not change processing times, i.e, $H_j(t)$ and $G_2(t)$ are degenerate distributions. Take $G_1(t)$ as any arbitrary distribution. Due dates also do not change. An optimal solution to this newly constructed *1 | X_j ~ $H_j(t)$; brkdwn: U ~ $G_1(t)$, D ~ $G_2(t)$ | RM2* instance is also an optimal solution to the original $1||\sum_j T_j$ instance. $1||\sum_j T_j$ is known to be $\mathcal{NP}$-hard (Du and Leung, 1990) and the result follows. □

Theorem 3.3 (Dominance Rule). *Consider 1 | $X_j \sim H_j(t)$ | RM2 problem. For any two jobs i and j if $X_i \leq_{st} X_j$ and $d_i \leq d_j$, then there exists an optimal sequence in which job i precedes job j.*

Proof. The proof is by an interchange argument. Let *S* be an optimal sequence in which job *j* precedes job *i*. Consider swapping job *i* and job *j* but do not touch the other jobs. Let *S′* be the newly obtained sequence. We compare *RM2*(*S*) with *RM2*(*S′*). Nothing changes for the jobs that precede job *j* or that succeed job *i* in *S*. Consider a job that succeeds job *j* but precedes job *i* in *S*, say job *k*. Let BS_k be the index set of jobs that precedes *k* and succeeds *j* in *S*. Let $C_k(S)$ denote the realized completion time of job *k* in *S*, and $C_k(S')$ denote the same in *S′*. Finally let T_1 be the time that job *j* starts its processing and T_2 be the time that job *i* finishes its processing in *S*. We have

$$C_k(S) = T_1 + X_j + \sum_{m \in BS_k} X_m + X_k \text{ and } C_k(S') = T_1 + X_i + \sum_{m \in BS_k} X_m + X_k .$$

Since $X_i \leq_{st} X_j$, we have $C_k(S') \leq_{st} C_k(S)$ by Proposition 3.1 and

$\max\{C_k(S') - d_k, 0\} \leq_{st} \max\{C_k(S) - d_k, 0\}$ by Proposition 3.2.

This leads $E[\max\{C_k(S') - d_k, 0\}] \leq_{st} E[\max\{C_k(S) - d_k, 0\}]$. Hence swapping cannot increase the expected tardiness of a job in between. Now let us consider jobs *i* and *j* themselves. Let ΔT_j be the increase in job *j*'s tardiness because of the interchange and similarly ΔT_i be the decrease in the tardiness of job *i*. We have

$$\Delta T_j = \begin{cases} T_2 - (T_1 + X_j) & \text{if} \quad d_j \leq T_1 + X_j \\ T_2 - d_j & \text{if} \quad T_1 + X_j \leq d_j \leq T_2 \\ 0 & \text{if} \quad d_j \geq T_2 \end{cases}$$

and

$$\Delta T_i = \begin{cases} T_2 - (T_1 + X_i) & \text{if} \quad d_i \leq T_1 + X_i \\ T_2 - d_i & \text{if} \quad T_1 + X_i \leq d_i \leq T_2 \\ 0 & \text{if} \quad d_i \geq T_2 \end{cases}$$

Just for now ignore the cases where $d_i \geq T_2$ or $d_j \geq T_2$. Then we have $\Delta T_j = T_2 - \max\{T_1 + X_j, d_j\}$ and $\Delta T_i = T_2 - \max\{T_1 + X_i, d_j\}$. Since $X_i \leq_{st} X_j$ and $d_i \leq d_j$, by coupling we have $\max\{T_1 + X_i, d_j\} \leq_{st} \max\{T_1 + X_j, d_j\}$. Therefore,

$E[\Delta T_i] - E[\Delta T_j] = E[\max\{T_1 + X_j, d_j\}] - E[\max\{T_1 + X_i, d_i\}] \geq 0$ and the interchange cannot degrade the objective function. Now let us examine the cases we have ignored. If $d_j \geq T_2$ then $\Delta T_j = 0$ and the interchange cannot lead to a worse solution, because the expected tardiness of job j does not increase and that of job j may possibly decrease. In the other case we have ignored, $d_i \geq T_2$, and since $d_i \leq d_j$ we must also have $d_j \geq T_2$. In that case, $\Delta T_i = \Delta T_j = 0$ and the interchange affects nothing. We conclude that S' is also an optimal sequence and this concludes the proof. □

Corollary 3.3. *Consider 1 | X_j ~ exponential(λ_j) | RM2 problem. If due dates are agreeable, i.e., if the earliest due date first (EDD) and SEPT sequences are the same, the EDD sequence is optimal.*

Corollary 3.4. *SEPT order gives an optimal solution for 1 | X_j ~ exponential(λ_j); $d_j = d$ | RM2.*

Note that if the processing times are exponentially distributed, Theorem 3.3 can be extended to include arbitrary machine breakdowns. As a result, Corollaries 3.3 and 3.4 are also still valid in the presence of machine breakdowns. For the proofs of the last two corollaries and the inclusion of the breakdown process, we refer the reader to Pinedo (2002), Section 10.4.

3.5 Stability

Recall that a stable schedule is one that should not deviate much from the initial schedule. The deviation is generally measured in terms of the differences between the job completion times in the initial and realized schedules. Hence, a typical stability measure is a non-decreasing function of the deviation of job completion times. We use

three stability measures: 1) the expected sum of squares of job completion time differences between the initial and realized schedules (*SM1*) 2) the sum of the variances of the realized completion times (*SM2*) 3) the expected absolute job completion time differences between the initial and realized schedules (*SM3*). *SM3* was already available in the literature. *SM1* and *SM2*, however, are proposed for the first time in this thesis.

3.5.1 Stability Measure 1 (SM1) and Stability Measure 2 (SM2)

Recall that *SM1* is the expected sum of squares of job completion time differences between the initial and realized schedules. That is, $SM1 = E[\sum_{i=1}^{n}(C_i^d - C_i)^2]$, where C_i^d is the deterministic completion time of job *i* without taking machine breakdowns or processing time variability into account.

A scheduler who is aware of the fact that initial schedules will inevitably deviate due to random disruptions can prepare his/her secondary plans according to expected completion times rather than deterministic completion times. In this case, a reasonable stability measure can be

$$SM2 = E[\sum_{i=1}^{n}(E[C_i] - C_i)^2] = \sum_{i=1}^{n} Var[C_i].$$

Theorem 3.4 (SVPT (Smallest Variance of Processing Time first) Optimality). *If* $Var[X_j] \leq Var[X_{j+1}]$ *for* $j = 1,\ldots,n-1$, *the job sequence* $\{1,\ldots,n\}$ *is an optimal solution to* $1 \mid X_j \sim H_j(t) \mid$ *SM1(SM2) problem. In other words, the SVPT rule gives an optimal solution.*

Proof. The proof is by contradiction. Let *S* be an optimal sequence but assume that there exists a pair of adjacent jobs *i* and *j* such that $Var[X_i] > Var[X_j]$ and job *j* succeeds job *i* in *S*. Now consider a sequence *S′*, obtained from *S* by swapping the positions of jobs *i* and *j*. We compare *SM1*(*S*) and *SM1*(*S′*). We may ignore the jobs

other than *i* and *j* in this comparison, since nothing changes for them. Let their contribution to the objective function be *c*. Let *T* denote the sum of the processing times of the jobs that precede *i* in *S*. We have

$$\begin{aligned} SM1(S) &= E[(T + X_i - E[T + X_i])^2] + E[(T + X_i + X_j - E[T + X_i + X_j])^2] + c \\ &= Var[T + X_i] + Var[T + X_i + X_j] + c \end{aligned}$$

and

$$\begin{aligned} SM1(S') &= E[(T + X_j - E[T + X_j])^2] + E[(T + X_i + X_j - E[T + X_i + X_j])^2] + c \\ &= Var[T + X_j] + Var[T + X_i + X_j] + c \end{aligned}$$

Hence, $SM1(S) - SM1(S') = Var[X_i] - Var[X_j]$. Since $Var[X_i] > Var[X_j]$, $SM1(S) > SM1(S')$. That is, there is a strict improvement in the objective function after the interchange. This contradicts the fact that *S* is an optimal solution. □

The result for *SM2* is actually a corollary to the above proof since the measures are equivalent in the case of no breakdowns.

Corollary 3.5. *SEPT* solves *1 | X_j ~ exponential(λ_j) | SM1(SM2)* optimally.

Theorem 3.5 (SEPT Optimality). *If* $E[X_i] > E[X_j]$ *implies* $Var[X_i] \geq Var[X_j], \forall (i, j)$ *then*

1 | X_j ~ $H_j(t)$; brkdwn: U ~ exponential(λ), D ~ $G_2(t)$ | SM1(SM2) is solved optimally by the SEPT rule.

Proof. The proof is again by contradiction. Let *S* be an optimal sequence but assume that there exists a pair of adjacent jobs *i* and *j* such that $E[X_i] > E[X_j]$ and job *j* succeeds job *i* in *S*. Now consider a sequence *S'*, obtained from *S* by swapping the positions of jobs *i* and *j*. We compare *SM1(S)* and *SM1(S')*. We may ignore the jobs other than *i* and *j* in this comparison, since nothing changes for them. Let their contribution to the objective function be *c*. Let BS_i denote the index set of jobs that precedes job *i* in *S*. Let $A_k = Y_k - E[X_k]$ for each job index *k*. We have

$$\begin{aligned} SM1(S) &= E[(\sum_{m\in BS_i} Y_m + Y_i - E[\sum_{m\in BS_i} X_m] - E[X_i])^2] + E[(\sum_{m\in BS_i} Y_m + Y_i + Y_j \\ &\quad - E[\sum_{m\in BS_i} X_m] - E[X_i] - E[X_j])^2] + c \\ &= E[(A_i + \sum_{m\in BS_i} A_m)^2] + E[(A_i + A_j + \sum_{m\in BS_i} A_m)^2] + c \end{aligned}$$

and similarly $SM1(S') = E[(A_j + \sum_{m\in BS_i} A_m)^2] + E[(A_i + A_j + \sum_{m\in BS_i} A_m)^2] + c$.

Then

$$\begin{aligned} SM1(S) - SM1(S') &= E[(A_i + \sum_{m\in BS_i} A_m)^2] - E[(A_j + \sum_{m\in BS_i} A_m)^2] \\ &= E[A_i^2 + (\sum_{m\in BS_i} A_m)^2 + 2A_i \sum_{m\in BS_i} A_m] \\ &\quad - E[A_j^2 + (\sum_{m\in BS_i} A_m)^2 + 2A_j \sum_{m\in BS_i} A_m] \\ &= E[A_i^2] - E[A_j^2] + 2E[A_i - A_j]E[\sum_{m\in BS_i} A_m] \end{aligned}$$

The last line is obtained by using the fact that A_k's are independent, since X_k's and Y_k's are independent. Note that $E[A_i] = E[Y_i] - E[X_i] = \lambda r E[X_i]$ and

$$\begin{aligned} E[A_i^2] &= E[(Y_i - E[X_i])^2] \\ &= E[Y_i^2 + (E[X_i])^2 + 2Y_i E[X_i]] \\ &= E[Y_i^2] + (E[X_i])^2 + 2E[Y_i]E[X_i] \\ &= Var[Y_i] + (E[Y_i])^2 + (E[X_i])^2 + 2E[Y_i]E[X_i] \\ &= \lambda(v + r^2)E[X_i] + (1 + \lambda^2 r^2)Var[X_i] + (1 + \lambda r)^2 (E[X_i])^2 \\ &\quad + (E[X_i])^2 + 2(1 + \lambda r)(E[X_i])^2 \\ &= \lambda(v + r^2)E[X_i] + (1 + \lambda^2 r^2)Var[X_i] + (\lambda^2 r^2 + 4\lambda r + 4)(E[X_i])^2 \end{aligned}$$

Then we have

$$\begin{aligned} SM1(S) - SM1(S') &= \lambda(Var[R] + r^2)(E[X_i] - E[X_j]) + (1 + \lambda^2 r^2)(Var[X_i] - Var[X_i]) \\ &\quad + (\lambda^2 r^2 + 2\lambda r + 4)((E[X_i])^2 - (E[X_j])^2) \\ &\quad + 2\lambda r(E[X_i] - E[X_j])E[\sum_{m\in BS_i} A_m] \end{aligned}$$

Since $E[X_i] > E[X_j]$ and $Var[X_i] \geq Var[X_j]$ this difference is strictly positive, which contradicts with the optimality of *S*.

The proof can be done with the same interchange argument for *SM2*. For *SM2* measure we have,

$$\begin{aligned} SM2(S) &= Var[\sum_{m\in BS_i} Y_m + Y_i] + Var[\sum_{m\in BS_i} Y_m + Y_i + Y_j] \\ SM2(S') &= Var[\sum_{m\in BS_i} Y_m + Y_j] + Var[\sum_{m\in BS_i} Y_m + Y_i + Y_j] \\ SM2(S) - SM2(S') &= Var[Y_i] - Var[Y_i] \\ &= \lambda(Var[R] + r^2)(E[X_i] - E[X_j]) \\ &\quad + (1 + \lambda^2 r^2)(Var[X_i] - Var[X_j]) \end{aligned}$$

Since $E[X_i] > E[X_j]$ and $Var[X_i] \geq Var[X_j]$ this difference is strictly positive, which contradicts with the optimality of *S*. □

Corollary 3.6. *SEPT* solves
1 | X_j ~ exponential(λ_j); brkdwn: U ~ exponential(λ), D ~ $G_2(t)$ | SM1(SM2) optimally.

1 | X_j ~ $H_j(t)$; brkdwn: U ~ $G_1(t)$, D ~ $G_2(t)$ | SM1(SM2) is analytically intractable in the general case.

3.5.2 Stability Measure 3 (SM3)

SM3 is the expected absolute job completion time differences between the initial and realized schedules.

$$SM3 = E[|C_i^d - C_i|]$$

This kind of measuring of the deviation between two schedules is first proposed by Wu *et al.* (1993).

Theorem 3.6 (SPT Optimality). *SPT solves*
1 | p_j, brkdwn: U ~ exponential(λ), D ~ $G_2(t)$ | SM3 optimally where p_j denotes the deterministic processing time of job j.

Proof. The proof is again by contradiction. Let *S* be an optimal sequence but assume that there exists a pair of adjacent jobs *i* and *j* such that $p_i > p_j$ and job *j* succeeds job *i* in *S*. Now consider a sequence *S′*, obtained from *S* by swapping the positions of jobs *i* and *j*. We compare *SM2*(*S*) and *SM2*(*S′*). We may ignore the jobs other than *i* and *j* in this comparison, since nothing changes for them. Let their contribution to the objective function be *c*. Let BS_i denote the index set of jobs that precedes job *i* in *S*. We have

$$\begin{aligned} SM3(S) &= E[|\sum_{m\in BS_i} Y_m + Y_i - \sum_{m\in BS_i} p_m - p_i|] \\ &\quad + E[|\sum_{m\in BS_i} Y_m + Y_i + Y_j - \sum_{m\in BS_i} p_m - p_i - p_j|] + c \end{aligned}$$

$$\begin{aligned} &= E[\sum_{m\in BS_i} Y_m + Y_i - \sum_{m\in BS_i} p_m - p_i] \\ &\quad + E[\sum_{m\in BS_i} Y_m + Y_i + Y_j - \sum_{m\in BS_i} p_m - p_i - p_j] + c \end{aligned}$$

$$\begin{aligned} SM3(S') &= E[\sum_{m\in BS_i} Y_m + Y_j - \sum_{m\in BS_i} p_m - p_j] \\ &\quad + E[\sum_{m\in BS_i} Y_m + Y_i + Y_j - \sum_{m\in BS_i} p_m - p_i - p_j] + c \end{aligned}$$

$$\begin{aligned} SM3(S) - SM3(S') &= E[Y_i - Y_j] - (p_i - p_j) \\ &= \lambda r(p_i - p_j) \end{aligned}$$

Since $p_i > p_j$ the improvement in objective function is strictly positive, which contradicts with the optimality of *S*. □

1 | X_j ~ $H_j(t)$; brkdwn: U ~ $G_1(t)$, D ~ $G_2(t)$ | SM3 is analytically intractable in the general case.

Theorems 3.1-3.6 and their corollaries can be summed up with a single principle for the single machine scheduling of jobs with processing time uncertainty: other things being equal, "shorter" (i.e., with a stochastically smaller processing time) and "safer" (i.e., with a smaller variance of processing time) jobs are to be scheduled first to optimize robustness and stability, respectively. Although this principle may commonly be used in everyday life, our results on the validity of the mentioned principle seem to be of interest for managerial purposes.

The results are summarized in Tables 3.1 and 3.2.

3.6 A Branch-and-Bound Algorithm for *1 | X_j ~ $H_j(t)$ | RM2*

In this section, we focus on the *1 | X_j ~ $H_j(t)$ | RM2* problem because of two reasons. First, the total tardiness performance measure is popular and frequently used in practice.

Table 3.1. Analytically Tractable Cases; Robustness

Problem	**Algorithm**	**Theorem/ Corollary**
*1	X_j ~ $H_j(t)$; brkdwn: U ~ $G_1(t)$, D ~ $G_2(t)$	RM1*	*SSPT*	Theorem 3.1
*1	X_j ~ exponential(λ_j); brkdwn: U ~ $G_1(t)$, D ~ $G_2(t)$	RM1.*	*SEPT*	Corollary 3.1
*1	X_j ~ $H_j(t)$	RM1.*	*SEPT*	Corollary 3.2
*1	X_j ~ $H_j(t)$	RM2*	*Dominance Rule*	Theorem 3.3
*1	X_j ~ exponential(λ_j)	RM2*	*EDD if EDD and SEPT sequences are the same*	Corollary 3.3
*1	X_j ~ exponential(λ_j); d_j = d	RM2.*	*SEPT*	Corollary 3.4

Table 3.2. Analytically Tractable Cases; Stability

Problem	**Algorithm**	**Theorem/ Corollary**
1 | X_j ~ $H_j(t)$ | SM1(SM2)	*SVPT*	Theorem 3.4
1 | X_j ~ exponential(λ_j) | SM1(SM2)	*SEPT*	Corollary 3.5
1 | X_j ~ $H_j(t)$; brkdwn: U ~ exponential(λ), D ~ $G_2(t)$ | SM1(SM2)	*SVPT if* $E[X_i] > E[X_j]$ *implies* $Var[X_i] \geq Var[X_j], \forall(i, j)$	Theorem 3.5
1 | X_j ~ exponential(λ_j); brkdwn: U ~ exponential(λ), D ~ $G_2(t)$ | SM1(SM2)	*SEPT*	Corollary3. 6
1 | p_j,brkdwn: U ~ exponential(λ), D ~ $G_2(t)$ | SM3	*SPT*	Theorem 3.6

Second, we have a dominance rule (Theorem 3.3) that can be effectively used in a branch-and-bound algorithm to keep the size of the search tree manageable.

The algorithm developed in this section is for the problems where the processing time distributions of any two jobs are stochastically comparable. Typical examples are normal distribution with a common coefficient of variation (c_v), Gamma distribution with the same scale parameter, and Poisson distribution. For all these distributions, ordering in the expected value corresponds to ordering in the stochastic sense. Moreover, the job completion times in any sequence also have the same type of distributions as the processing time distributions, i.e., they belong to the same family.

We should be very careful when processing time distributions are normal with a common coefficient of variation because stochastic comparability is only valid for the nonnegative part of the distributions. Thus, the probability of having negative processing times should be negligibly small (c_v < 1/3) for a satisfactory performance of the algorithm.

In the proposed B&B algorithm, we develop the schedules progressively in the forward direction. At level *k* of the branch-and-bound tree, jobs in the first *k* positions are specified. We use the dominance rule in Theorem 3.3 during the branching process. The initial upper bound is taken as the minimum expected total tardiness value of the SPT, EDD and ATC solutions. The upper bound of each node is taken as

the expected total tardiness of the EDD completion of that node. If the global upper bound is greater than the upper bound of a node, it is updated. There are two lower bounds considered: a loose one and a tight one. These are explained in the next two theorems. If the lower bound of a node is greater than the global upper bound, it is pruned. We use a most-promising-node-first exploration strategy, that is, the node with the lowest upper bound value is branched first.

Theorem 3.7 (Lower Bound 1). *Consider the problem 1 | $X_j \sim H_j(t)$ | RM2. Arrange the due dates in non-decreasing order and assign them to the jobs arranged in a stochastic non-decreasing order of processing times (assuming all jobs can be ordered stochastically). The optimal expected total tardiness value of this new problem P1 is a lower bound on the optimal objective value of the original problem 1 | $X_j \sim H_j(t)$ | RM2.*

Proof. The proof is by an interchange argument. The optimal solution to *P1* is an EDD sequence by Theorem 3.3. We now show that its objective function value is a lower bound on the optimal objective function value of the original problem. We begin by an optimal solution S^* of the original problem and convert it to an optimal solution of *P1*. The procedure is as follows:

Step 1 Consider every adjacent job pair. If a job with a greater due date precedes a job with a smaller due date, swap their due dates but not their positions; just assign the due date of the former job to the latter job and vice versa. Continue in this fashion until all due dates are in non-decreasing order. Each swap of the due dates results in a possible decrease in the expected total tardiness of the schedule but never an increase.

Step 2 Take the resultant schedule of Step 1 as the input and process it in the same way as in Step 1. The only difference is, instead of due-dates, compare and exchange the processing times of the adjacent job pairs if necessary.

The resulting schedule is optimal for *P1* and its objective function value is a lower bound on that of the original problem. The deterministic version of this lower bound is developed by Chu (1992). □

Della Croce, Tadei, Baracco, and Grosso (1998) propose another lower bound for the deterministic problem. Here, we extend this lower bound to the stochastic problem as follows:

Theorem 3.8 (Lower Bound 2). *Consider the 1 | $X_j \sim H_j(t)$ | RM2 problem. Relabel the jobs according to the non-decreasing stochastic order of their processing times (assuming all jobs can be ordered stochastically). That is, the job with the stochastically smallest processing time is job 1, and with the stochastically largest processing time is job n. Split the job set J into two subsets $J_1 = \{1,\ldots,l\}$ and $J_2 = \{l+1,\ldots,n\}$, where $l = \lfloor n/2 \rfloor$. For each subset, separately arrange the due dates in non-decreasing order and assign them to the jobs arranged in a stochastic non-decreasing order of processing times. The optimal expected total tardiness value of this new problem P2 is a lower bound on that of the original problem 1 | $X_j \sim H_j(t)$ | RM2.*

The proof of the theorem basically involves the same interchange argument as in the proof of Theorem 3.7. The only difference is that we apply Steps 1 and 2 separately to the jobs in subsets J_1 and J_2. That is, at each pass of Step 1 or Step 2, we examine successive jobs (not necessarily adjacent) that belong to the same subset. At the end, we obtain a feasible schedule to *P2*, whose expected total tardiness is no greater than the optimal objective value of the original problem. The expected total tardiness value of an optimal solution to *P2* is possibly even less, so it is a lower bound for the original problem. The optimal solution to *P2* can be found in polynomial time. For two solution procedures, each with $O(n^2)$ time complexity, the reader can refer to Della Croce *et al.* (1998).

3.7 A Beam-Search Algorithm for Other Intractable Problems

The proposed branch-and-bound algorithm relies on Theorem 3.3 as a dominance rule and Theorems 3.7 and 3.8 as lower bounds. These theorems are valid under the

assumption that for any two jobs, their processing times are stochastically comparable. Also, machine breakdowns are not considered. In this section, we develop a beam-search algorithm that can be used with any processing time distribution and any objective function (*RM1*, *RM2*, *SM1*, *SM2*, or *SM3*) and that can also handle a general machine breakdown/repair process.

Beam search is an approximate branch-and-bound method which operates on a search tree. BS has been used to solve combinatorial optimization problems for the last two decades. There are several successful applications to job-shop scheduling and flexible manufacturing systems (FMS) scheduling problems with static and deterministic assumptions (Sabuncuoglu and Karabuk (1998) and Sabuncuoglu and Bayiz (1999)). Generally speaking, BS is similar to a breadth-first search as it progresses level by level without backtracking. However, unlike breadth first, only the best β (*beam width*) promising nodes are kept for further branching at any level. The potential promise of each node is determined by a *global evaluation function*, which typically estimates the minimum total cost of the best solution obtained from the partial schedule represented by the node.

In a BS implementation, the beams may progress independently (i.e., at all levels other than level 1, each of β promising nodes has a different ancestor), but in our implementation, we use dependent beams (i.e., at each level, all the descendants are evaluated and the best β of them are chosen without paying attention to their ancestors). Figure 3.1 illustrates a hypothetical example with $\beta = 2$. Specifically, we first complete the partial schedule that the node represents according to the objective function in use. If the objective function is *RM1*, the schedule is completed according to the SEPT rule. Similarly, ATC is used for *RM2*, and SVPT is used for *SM1*, *SM2*, or *SM3*. We then simulate the resulting schedule 10 times. The average of these objective function values is taken as the global evaluation function value. The simulations are done with the help of a simple discrete-event simulation model coded in C++ language. First, the processing times of the jobs are generated according to their respective probability distributions. After that, the machine uptimes and downtimes are generated and inserted into their proper positions in the schedule. Finally, the realized job completion times are obtained and used for the performance measure calculations.

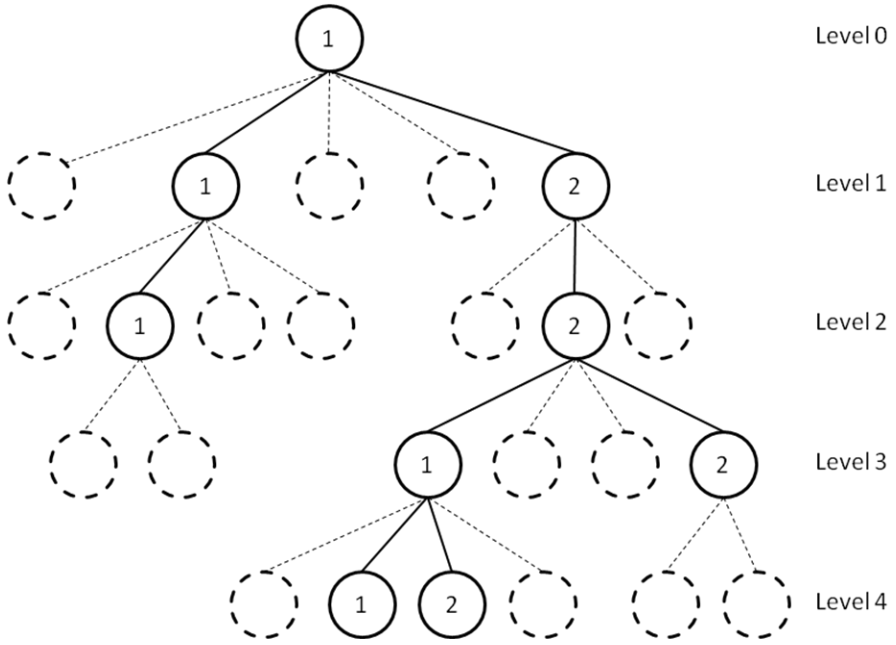


Figure 3.1. Beam-Search Example with $\beta = 2$

3.8 Computational Experiments

The performance of the proposed algorithms is measured on a non-dedicated Linux box with dual AMD Opteron 2.6GHz CPUs and 2GBs of physical memory. The codes

are written in C++ language. The data generation scheme, initially proposed by Fisher (1976), is explained in the next section.

3.8.1 Test Problems and Beam-Search Parameters

The problem instances of varying degrees of difficulty are generated by means of two factors: tardiness factor (*TF*) and range of due dates (*DR*). For each problem, first the processing time means are generated from a uniform distribution with parameters (1, 100). Then the due dates are generated from a uniform distribution, which depends on

the sum of the processing time mean (P), and on R and T. The due date distribution is uniform over [$P(1 - TF - DR / 2)$, $P(1 - TF + DR / 2)$]. The values of TF and DR are selected from {0.2, 0.4, 0.6, 0.8} and {0.2, 0.4, 0.6, 0.8, 1.0}, respectively. This yields 20 combinations of TF, DR for each problem size. The number of jobs n is selected from the set {10, 20, 30, 40, 50, 75, 100}. 10 different instances are solved for each setting of n, TF, DR, which gives 200 instances for each choice of n. For the B&B algorithm, we solve problems up to the size of 10 ($n = 10$). For the beam-search algorithm, we solve all problem sizes for each objective function (*RM1*, *RM2*, *SM1*, *SM2*, and *SM3*). Table 3.3 summarizes the experimental settings.

We call each combination of n, TF, and DR a problem class. We also assign each problem class a code name: *probxyz*. Here x, y, z are the levels of n, TF, and DR factors, respectively. For example, prob231 is the problem class in which n = 20, TF = 0.6, and DR = 0.2.

The beam width is taken as 4. Recall that the proposed BS algorithm employs simulation as the global evaluation function. During simulation runs, we use Gamma distribution as a busy-time distribution with a shape parameter of 0.7, and a scale parameter to be specified. We use Gamma distribution with a shape parameter of 1.4 for the down-time distribution, as recommended by Law and Kelton (2000). The scale parameter of the busy-time distribution is arranged so that the mean is 300. Similarly, the scale parameter of the down-time distribution is arranged so that the mean is 50.

Table 3.3. Experimental Environment

Processing time mean ($E[X_j]$)	U[1, 100]
Number of jobs (n)	10, 20, 30, 40, 50, 75, 100
Due dates (d_j)	U[$P(1 - TF - DR / 2)$, $P(1 - TF + DR / 2$], where P is the sum of processing time means TF in {0.2, 0.4, 0.6, 0.8} DR in {0.2, 0.4, 0.6, 0.8, 1.0}

3.8.2 Evaluation of the Algorithms for $1 \mid X_j \sim H_j(t) \mid RM2$

Recall that the branch-and-bound algorithm is developed for the *1 | X_j ~ $H_j(t)$ | RM2* problem (i.e., there are no machine breakdowns) and the processing time distributions of any two jobs can be stochastically compared. We take the processing time distribution of each job as the Gamma distribution, with a scale parameter of 2. The shape parameters are arranged such that the mean processing times equal the previously generated values (see Section 3.8.1). Only 200 10-job problems are solved because of the computational time limitations. Each problem instance is solved two times, once using Lower Bound 1 (loose) and once using Lower Bound 2 (tight). Table 3.4 presents the results. In Table 3.4, better CPU time is marked with an asterix (*) for each problem class. Figures 3.2 and 3.3 are prepared to observe the effects of *TF* and *DR* clearly.

We observe two things: 1) Generally, as *TF* and *RD* increase, the problems get easier and require less computational time to solve (i.e., the problems with loose due dates are harder to solve), and 2) the extra computational time required for calculating a tight lower bound pays off for hard problem classes, but this is not worth the effort for easy problem classes.

Table 3.4. Results for Branch and Bound

Problem	**CPU Time**		**Objective**	**Problem**	**CPU Time**		**Objective**
	LOOSE	**TIGHT**			**LOOSE**	**TIGHT**	
prob111	1962.23	1833.26*	75.64	prob131	292.81*	432.88	487.65
prob112	778.20*	1255.01	36.72	prob132	205.67*	276.43	593.44
prob113	318.40*	507.28	27.16	prob133	133.51*	171.97	596.93
prob114	557.84*	671.69	34.41	prob134	346.47*	417.92	487.82
prob115	479.59*	670.11	23.18	prob135	646.03	575.50*	707.64
prob121	849.89	816.65*	305.25	prob141	65.91*	144.03	1129.50
prob122	800.41*	1055.06	210.04	prob142	68.32*	148.94	1351.72
prob123	956.70	695.33*	199.16	prob143	91.42*	192.63	1410.66
prob124	323.33*	472.76	124.38	prob144	141.06*	196.17	1140.40
prob125	645.20	593.21*	124.64	prob145	78.82*	93.82	1297.68

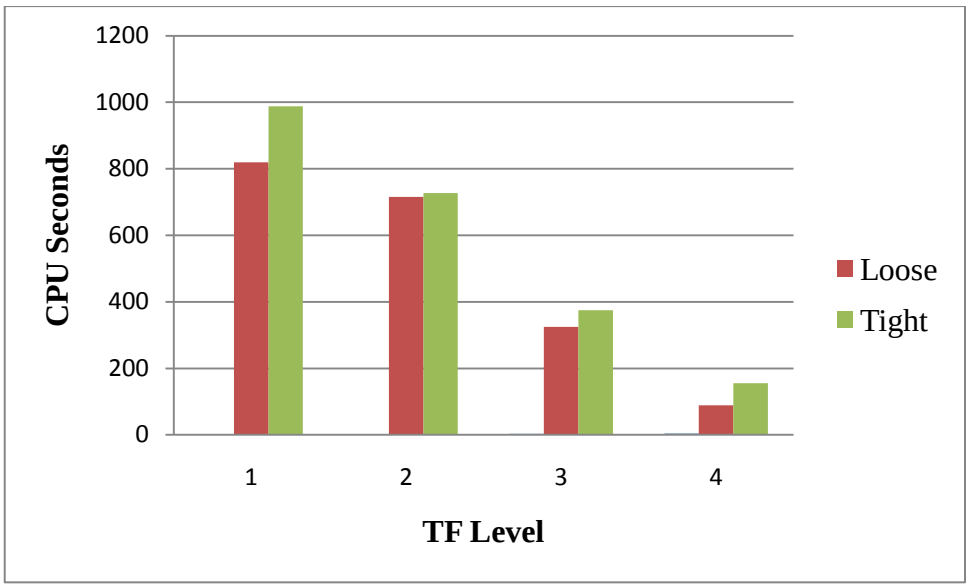


Figure 3.2. Effect of TF Level on CPU Seconds

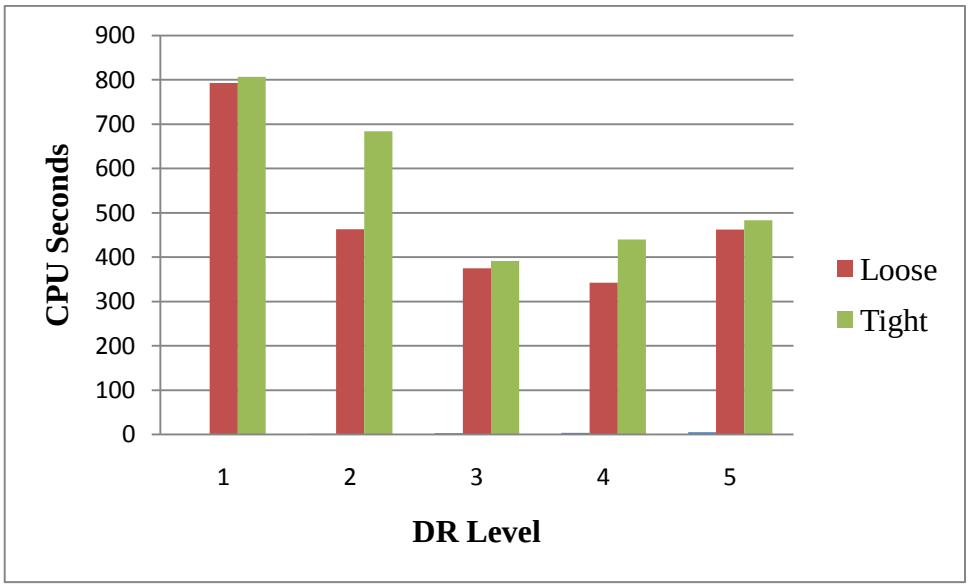


Figure 3.3. Effect of DR Level on CPU Seconds

Tables 3.5 and 3.6 present the average number of pruned nodes due to the dominance rule for loose and tight lower bounds, respectively. We can observe that the dominance rule works quite effectively.

For example, on average, 7.1 nodes are pruned due to the dominance rule among 10 nodes in level 1. Among (10-7.1) x 9 = 26.1 nodes in level 2, the dominance rule prunes 15.1 and 13.7 for the algorithms with loose and tight lower bounds, respectively.

We also observe that the dominance rule prunes fewer nodes in each level for the algorithm with the tight lower bound, but this is expected because for this case, more nodes are pruned due to their upper and lower bound comparisons.

Table 3.5. Performance of Dominance Rule, Loose Lower Bound

	Nodes Pruned – Loose								
Problem	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**	**Level 6**	**Level 7**	**Level 8**	**Level 9**
prob111	7.5	11.3	29.9	78.8	196.5	462.2	800.4	758.9	0.1
prob112	7.1	8.7	25.1	67.0	160.2	296.5	317.4	195.6	0.0
prob113	7.6	12.1	21.7	44.9	89.2	135.8	141.9	70.2	0.2
prob114	7.0	12.5	29.0	69.8	157.8	200.1	183.6	65.9	0.0
prob115	6.3	13.9	38.0	93.0	183.4	242.0	147.9	38.3	0.0
prob121	7.2	17.3	38.9	95.4	202.5	330.3	386.7	167.7	0.1
prob122	6.9	18.2	47.0	109.5	226.1	414.9	380.5	196.8	0.1
prob123	7.1	17.3	43.7	103.8	231.4	420.0	447.8	154.8	0.1
prob124	7.3	16.3	36.2	66.3	115.9	150.0	103.0	51.0	0.0
prob125	6.8	16.7	40.8	95.4	201.6	309.0	233.8	59.4	0.0
prob131	7.0	16.7	40.8	95.5	176.8	199.6	59.7	4.2	0.0
prob132	7.6	15.8	29.6	59.8	93.6	100.0	50.9	11.4	0.0
prob133	8.0	13.7	21.5	40.9	61.1	45.8	15.9	2.5	0.0
prob134	7.0	16.9	38.4	92.1	182.5	222.3	115.5	30.7	0.0
prob135	6.6	19.4	46.7	98.6	156.2	161.4	137.3	27.3	0.3
prob141	6.6	18.7	37.6	50.3	46.5	19.4	8.1	5.4	0.7
prob142	7.1	14.5	26.0	28.4	21.6	10.5	5.9	3.5	0.0
prob143	7.5	11.5	17.4	26.7	29.1	19.0	6.5	3.9	0.0
prob144	6.7	17.9	35.1	59.2	61.2	41.1	21.7	5.5	0.5
prob145	7.5	13.3	20.6	32.4	41.5	43.5	22.8	3.3	0.0
Average	7.1	15.1	33.2	70.8	131.7	191.2	179.4	92.8	0.1

Table 3.6. Performance of Dominance Rule, Tight Lower Bound

	Nodes Pruned – Tight								
Problem	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**	**Level 6**	**Level 7**	**Level 8**	**Level 9**
prob111	7.5	10.0	17.9	49.3	131.9	310.0	340.6	192.6	0.0
prob112	7.1	8.7	20.6	52.1	126.6	234.5	277.9	126.2	0.0
prob113	7.6	10.9	20.5	38.9	63.2	110.3	87.4	12.8	0.2
prob114	7.0	12.5	20.4	43.0	87.3	126.4	125.4	19.8	0.0
prob115	6.3	12.0	32.5	64.8	106.6	73.4	43.1	0.0	0.0
prob121	7.2	16.0	33.6	74.0	133.2	135.9	76.9	16.0	0.1
prob122	6.9	17.2	41.4	84.9	159.0	179.2	140.5	47.9	0.1
prob123	7.1	13.9	32.4	63.5	102.6	79.0	57.8	3.0	0.1
prob124	7.3	13.0	24.3	41.1	56.3	75.9	32.5	19.8	0.0
prob125	6.8	13.2	30.9	61.6	91.3	92.7	41.6	11.6	0.0
prob131	7.0	15.5	36.6	78.6	120.5	62.9	20.0	3.3	0.0
prob132	7.6	15.8	28.4	44.5	61.5	38.7	14.7	5.2	0.0
prob133	8.0	13.0	17.2	24.8	26.5	16.4	8.7	2.4	0.0
prob134	7.0	16.9	30.0	55.8	80.8	54.9	34.5	7.9	0.0
prob135	6.6	17.2	32.0	48.5	61.0	54.9	21.1	3.4	0.3
prob141	6.6	18.0	34.7	42.0	36.7	17.7	8.1	5.4	0.7
prob142	7.1	13.1	21.3	19.4	15.9	7.9	5.9	3.5	0.0
prob143	7.5	10.0	15.8	21.3	22.9	14.4	5.1	3.9	0.0
prob144	6.7	16.6	27.6	38.3	27.4	13.6	12.4	5.3	0.5
prob145	7.5	11.2	15.9	18.2	20.3	12.9	10.3	1.6	0.0
Average	7.1	13.7	26.7	48.2	76.6	85.6	68.2	24.6	0.1

The same 200 problems are also solved by the proposed BS algorithm for comparison. The solutions obtained from the BS are evaluated by the exact objective function, which is also used in the branch-and-bound algorithm (i.e., the reported results are not simulation values). Table 3.7 summarizes the results. Optimal objective function values and minimum CPU times obtained from the branch-and-bound algorithm are also included in Table 3.7. The results indicate that the beam-search algorithm finds the optimal solution for 51 of the 200 problem instances. The deviation from the optimal values is under 2% for most of the problem classes.

A paired t-test with $\alpha = 0.05$ indicates that the differences in objective function are statistically significant for only the problem classes *prob122* and *prob132*.

Table 3.7. Branch and Bound vs. Beam Search

Problem	BS CPU Time	B&B CPU Time	BS Objective	Optimal Objective	Deviation from the optimal (%)
prob111	2.74	1833.26	77.50	75.64	2.46
prob112	2.93	778.20	37.18	36.72	1.26
prob113	2.55	318.40	30.42	27.16	12.01
prob114	2.65	557.84	34.88	34.42	1.34
prob115	2.71	479.59	23.47	23.18	1.26
prob121	2.86	816.65	307.22	305.25	0.65
prob122	2.68	800.41	212.92	210.04	1.37
prob123	2.80	695.33	207.62	199.16	4.25
prob124	2.80	323.33	128.84	124.38	3.58
prob125	2.61	593.21	125.41	124.64	0.62
prob131	2.22	292.81	489.71	487.65	0.42
prob132	2.93	205.67	599.64	593.44	1.04
prob133	3.20	133.51	601.64	596.93	0.79
prob134	2.63	346.47	492.31	487.82	0.92
prob135	2.71	575.50	709.03	707.64	0.20
prob141	2.57	65.91	1130.89	1129.50	0.12
prob142	2.99	68.32	1355.79	1351.72	0.30
prob143	3.01	91.42	1415.11	1410.66	0.32
prob144	2.78	141.06	1142.29	1140.40	0.17
prob145	2.94	78.82	1297.71	1297.68	0.00

We can conclude that the proposed beam search performs quite satisfactorily for the $1 \mid X_j \sim H_j(t) \mid RM2$ problem and, if computational time is an issue, it can be safely used to generate schedules instead of the exact algorithm.

We also compare the performances of the beam-search algorithm and the ATC dispatching rule. Table 3.8 presents a summary of the results. The objective function values reported in this table are the averages of the simulated total tardiness values of the schedules generated by the algorithms.

We observe that the beam-search algorithm performs better and all the differences are found to be significant by a paired t-test with $\alpha = 0.05$.

Table 3.8. Beam Search vs. ATC for RM2 No Breakdown; Summary

BEAM SEARCH			ATC Objective	Deviation (%)
# of Jobs	CPU Time	Objective		
10	1.22	527.01	547.48	3.88
20	10.31	1760.17	1804.69	2.53
30	34.47	3579.49	3652.44	2.04
40	83.63	6145.22	6253.40	1.76
50	164.13	9393.70	9562.67	1.80
75	556.58	20723.12	21088.92	1.77
100	1323.28	35926.39	36466.31	1.50

3.8.3 Evaluation of Proposed BS Algorithm for Other Intractable Problems with Machine Breakdowns

The performance of the proposed BS algorithm is evaluated by solving numerous problem instances for each objective function. Since *RM1, SM1, SM2,* and *SM3* are not due-date related performance measures, 10 instances from *probx11* (x = 1, …, 7) classes are used during the experiments, giving rise to 70 problem instances for each objective function. In other words, tardiness factor (*TF*) and range of due dates (*DR*) do not vary among test problems because they are irrelevant. For *RM2*, 10 instances from *probxyz* (x = 1, …, 7, y = 1, …, 4, z = 1, …, 5) classes are used, yielding a total of 1400 problem instances.

All problems include machine breakdown/repair. Since these problems are analytically intractable we do not know their optimal solutions. Thus, we compare the performance of the proposed BS algorithm to a priority dispatching rule for each objective function. The dispatching rule that is used depends on the objective function. For example, if the objective function is *RM1*, SEPT is used. Similarly, SVPT is used for the stability measures (*SM1*, *SM2*, or *SM3*). Note that *SEPT* is optimal for *RM1* and SVPT is optimal for *SM1*, *SM2*, or *SM3* under special conditions (see Theorem 3.1 and Theorems 3.4-3.8). However, we expect these dispatching rules to also perform well under more general conditions (even if the stated optimality conditions in Theorems 3.1 and 3.4-3.8 do not hold).

For *RM2*, three dispatching rules and three versions of the beam-search algorithm are considered. The first dispatching rule is ATC: at every time point *t* the machine becomes free, a priority index is calculated for each unscheduled job *j*, and the job with the highest priority is scheduled next. Note that the priority indices are calculated only at the deterministic completion times of the jobs. Additionally, two proactive versions of ATC, namely ProATC1 and ProATC2, are developed to incorporate the machine breakdown and repair information. In ProATC1, a job's processing time is inflated by the expected repair duration during the processing of that job. Specifically, the processing time for job *j* is taken as

$$p_j = E[X_j] + \frac{E[X_j]}{E[U]} E[D] = E[X_j](1 + \frac{E[D]}{E[U]}).$$

The priorities of the jobs are calculated using these new processing time values. In ProATC2, the time points where the priority indices are calculated are adjusted to include machine breakdowns. We anticipate a constant downtime period (*E[D]*) after every constant busy time period (*E[U]*). That is, time (*t)* is advanced by *E[D]* every time the machine stays up for *E[U]*. The beam-search algorithms under consideration are *classical BS*, *simulation-based BS*, and *proactive BS*. In classical BS, the global evaluation function is the regular total tardiness measure. At each level of the search tree, partial schedules in the nodes are completed by the ATC rule, and β nodes with the smallest total tardiness values are retained while the others are pruned permanently. Note that classical BS does not consider breakdowns or processing time variability. Simulation-based BS is like classical BS, except that it employs simulation as global evaluation function, therefore processing time variability and machine breakdowns are considered. In proactive BS, similar to simulation-based BS, the global evaluation function is based on simulation. The only difference is that in simulation-based BS, the partial schedules in the nodes are completed by the ATC rule before global evaluation, whereas in proactive BS they are completed by ProATC2.

To observe the effect of using simulation instead of surrogate measures, the same problem instances are also solved with a variant of the proposed BS algorithm (called *BS-M1*) for each objective function. The most frequently used surrogate measure in the literature is the average slack method developed by Leon *et al.* (1994).

This measure is developed for a job shop environment with the makespan measure. The measure depends on job *slacks*, which is defined as the amount of time that a job's processing can be delayed without increasing the makespan of the schedule. Since in this study we operate in a single machine environment with all jobs present at time t = 0, slacks for all jobs are zero and a slack-based measure cannot be applied. There are other surrogate measures that require inserting idle times, as in Mehta and Uzsoy (1998). Since our solution space is the class of non-delay schedules, these types of surrogate measures are not quite applicable. BS-M1 uses Method 1 surrogate measure in Goren and Sabuncuoglu (2008) to globally evaluate the nodes instead of simulation. Method 1 assumes that the machine fails after every constant busy time period of length $\lambda L + (1 - \lambda)U$, where λ is a real number between zero and one, and L and U are the 25th and 975th 1000-tiles of the busy-time distribution $G_1(t)$. It is also assumed that all repair activities last for a time period of length r, the expectation of the repair time distribution $G_2(t)$. Method 1 is developed for an environment where the job processing times are deterministic. To use it as a global evaluator, we further assume that the job processing times are deterministic and their values are equal to the expectations of the respective processing time distributions. To globally evaluate a node, the partial schedule at that node is first completed according to the SPT rule. Next, constant uptimes and downtimes are inserted and a new schedule that represents an approximate realization is obtained. Job completion times in this new schedule are used to calculate the performance measure of the node instead of simulation. The computational tests in Goren and Sabuncuoglu's correlation study (2008) indicate $\lambda = 0.6$ performs well. Since the same up- and down-time distributions are used in this study, the same λ value is also used.

To observe the impact of different repair time distributions on the performance of the proposed BS algorithm, the experiments are also conducted with an exponential repair time distribution (with the same mean) instead of Gamma.

During our tests, we take the processing time distributions as exponential except for *RM1*. We use normal distribution for *RM1* because the SEPT schedule would be already optimal if the processing times were exponentially distributed (see Theorem 3.1). For *RM1*, variances of the processing times are generated as uniformly distributed over [1, 100]. If a negative processing time value is generated during the simulations, it is simply ignored and generated again.

The simulations (both as a global evaluator in the BS algorithms and as an estimator of the resulting objective function value for all algorithms) during the experiments performed in this section are replicated 100 times instead of 10.

A summary of the results is given in Tables 3.9 and 3.10; the best objective function values are marked with an asterix (*) whereas the worst ones are marked with a '+' sign.

As can be seen in Tables 3.9 and 3.10, for the *RM1* performance measure the proposed BS and BS-M1 are competitive. For the case with Gamma repair time distribution, the proposed BS generally performs better, whereas BS-M1 performs the best for the case with exponential repair time distribution.

For all three stability measures, the proposed BS algorithm is significantly better than the corresponding dispatching rule or BS-M1. We observe that BS-M1 gets better with increasing problem sizes. Regardless of the repair time distribution, dispatching rules perform better than BS-M1 for small problems while BS-M1 performs better for larger problems.

We also observe that the differences between the performances of the alternative algorithms for *RM1* are relatively small compared to the other measures. The reason for such a good performance of SEPT for *RM1* is that the optimality conditions stated in Theorem 3.1 are mostly satisfied for *RM1* (except for stochastic comparability), whereas these conditions are not satisfied due to machine breakdown/repair for other measures and their respective theorems. This indicates that relaxing the stochastic comparability constraint is not as serious as relaxing the constraints on the machine breakdown/repair process. The summary of the results for *RM2* is given in Tables 3.11-3.14.

We make three main observations. First, the proactive approach does not always improve the performance of dispatching rules (in particular, ATC in our case) if it is not appropriately used. This is attested to by the better performance of traditional ATC over ProATC1. Our further investigation of this result indicates that how total repair time is distributed is important for the proactive use of dispatching rules.

Table 3.9. Comparison of Algorithms, Non-Due-Date Related, Gamma Repair Times

Objective Function	# of Jobs	Beam Search		Surrogate (BS-M1)		Dispatching Rule (SEPT/SVPT)	
		CPU Time	Objective	CPU Time	Objective	CPU Time	Objective
RM1	10	0.09	2,092.52	0.00	2,091.84*	0.00	2,103.06+
	20	0.77	9,116.38*	0.02	9,123.58	0.00	9,154.53+
	30	2.52	18,244.30	0.08	18,237.20*	0.00	18,430.70+
	40	6.11	32,474.30*	0.20	32,487.80	0.00	33,240.40+
	50	11.65	48,973.80*	0.36	48,992.00	0.00	50,259.40+
	75	39.23	110,345.00*	1.36	110,373.00	0.00	114,193.00+
	100	93.35	191,219.00	3.13	190,989.00*	0.00	199,043.00+
SM1	10	0.07	126,503.00*	0.00	276,603.00+	0.00	134,952.00
	20	0.57	643,599.00*	0.02	966,289.00+	0.00	697,516.00
	30	1.83	1,265,710.00*	0.07	1,411,240.00+	0.00	1,360,740.00
	40	4.41	2,505,680.00*	0.19	2,961,280.00	0.00	3,158,530.00+
	50	8.45	3,413,200.00*	0.34	4,090,790.00	0.00	4,684,200.00+
	75	28.43	8,049,690.00*	1.24	8,890,310.00	0.00	10,053,000.00+
	100	67.30	15,844,300.00*	2.80	16,313,500.00	0.00	22,206,400.00+
SM2	10	0.07	122,703.00*	0.00	270,075.00+	0.00	131,267.00
	20	0.58	608,366.00*	0.02	915,720.00+	0.00	648,840.00
	30	1.84	1,099,120.00*	0.08	1,224,390.00	0.00	1,249,160.00+
	40	4.43	2,180,560.00*	0.18	2,545,270.00	0.00	2,668,970.00+
	50	8.50	2,860,190.00*	0.34	3,382,500.00	0.00	3,692,810.00+
	75	28.60	6,112,700.00*	1.26	6,776,440.00	0.00	7,311,420.00+
	100	67.71	11,019,900.00*	2.81	11,458,700.00	0.00	15,095,700.00+
SM3	10	0.07	678.40*	0.00	1,101.34+	0.00	716.07
	20	0.58	2,280.15*	0.02	3,049.16+	0.00	2,400.81
	30	1.83	3,904.92*	0.08	4,372.56+	0.00	4,216.34
	40	4.42	6,150.28*	0.19	7,224.14+	0.00	7,043.67
	50	8.49	8,226.24*	0.34	9,508.64+	0.00	9,757.92
	75	28.43	15,374.30*	1.25	16,827.40	0.00	17,881.30+
	100	67.29	25,692.50*	2.88	26,264.00	0.00	31,096.80+

Table 3.10. Comparison of Algorithms, Non-Due-Date Related, Exponential Repair Times

Objective Function	# of Jobs	Beam Search		Surrogate (BS-M1)		Dispatching Rule (SEPT/SVPT)	
		CPU Time	Objective	CPU Time	Objective	CPU Time	Objective
RM1	10	0.08	2,137.29	0.00	2,135.48*	0.00	2,154.75+
	20	0.63	9,336.27	0.02	9,327.52*	0.00	9,483.38+
	30	2.07	18,852.60	0.08	18,839.60*	0.00	19,219.10+
	40	4.99	33,646.20	0.20	33,644.30*	0.00	34,648.70+
	50	9.50	50,851.00	0.36	50,780.70*	0.00	52,419.30+
	75	31.98	114,672.00	1.36	114,605.00*	0.00	120,134.00+
	100	75.94	198,884.00	3.11	198,684.00*	0.00	209,671.00+
SM1	10	0.06	150,470.00*	0.00	305,607.00+	0.00	154,277.00
	20	0.45	732,659.00*	0.02	1,073,150.00+	0.00	906,319.00
	30	1.44	1,512,100.00*	0.07	1,768,840.00+	0.00	1,765,760.00
	40	3.45	3,249,080.00*	0.19	3,832,140.00	0.00	4,009,570.00+
	50	6.62	4,509,910.00*	0.35	5,313,940.00	0.00	5,975,470.00+
	75	22.16	11,135,500.00*	1.24	12,238,900.00	0.00	13,105,900.00+
	100	52.29	23,435,700.00*	2.81	24,137,100.00	0.00	32,252,900.00+
SM2	10	0.06	142,760.00*	0.00	293,771.00+	0.00	146,447.00
	20	0.45	664,283.00*	0.02	981,248.00+	0.00	778,273.00
	30	1.45	1,208,060.00*	0.08	1,407,020.00	0.00	1,448,110.00+
	40	3.48	2,570,080.00*	0.20	3,008,500.00+	0.00	2,902,690.00
	50	6.68	3,349,290.00*	0.36	3,961,910.00	0.00	4,487,440.00+
	75	22.42	7,109,220.00*	1.26	7,870,520.00	0.00	8,597,210.00+
	100	52.95	13,175,600.00*	2.80	13,673,700.00	0.00	18,327,800.00+
SM3	10	0.05	733.99*	0.00	1,151.43+	0.00	749.51
	20	0.45	2,397.88*	0.02	3,188.48+	0.00	2,689.12
	30	1.45	4,163.81*	0.07	4,825.66+	0.00	4,718.05
	40	3.47	6,914.60*	0.19	8,127.66+	0.00	7,617.44
	50	6.63	9,464.42*	0.35	10,693.70	0.00	11,161.50+
	75	22.22	17,830.90*	1.23	19,591.90	0.00	21,767.20+
	100	52.40	30,896.10*	2.75	31,797.70	0.00	37,760.50+

Table 3.11. Dispatching Rules, RM2, Gamma Repair Times

# of Jobs	Dispatching Rule		
	ATC	ProATC1	ProATC2
	Objective	Objective	Objective
10	857.31	1,140.26	856.35
20	2,872.70	3,672.90	2,846.45
30	5,875.01	7,833.76	5,819.89
40	10,354.70	12,943.80	10,252.10
50	15,776.20	20,532.40	15,617.40
75	34,562.20	43,043.50	33,684.00
100	62,468.50	75,501.60	61,303.20

Table 3.12. Comparison of Algorithms, RM2, Gamma Repair Times

# of Jobs	Beam Search						Surrogate BS-M1	
	Classical BS		Simulation-based BS		Proactive BS			
	CPU Time	Objective	CPU Time	Objective	CPU Time	Objective	CPU Time	Objective
10	0.00	975.64	0.07	787.01	0.07	788.55	0.00	965.75
20	0.04	2,960.23	0.57	2,682.17	0.56	2,671.04	0.04	2,866.22
30	0.15	6,327.45	1.89	5,668.54	1.89	5,615.59	0.16	6,034.95
40	0.44	10,063.00	4.59	9,143.85	4.58	9,031.98	0.45	9,593.11
50	0.96	15,991.60	9.05	14,482.00	9.05	14,196.40	0.98	15,037.50
75	4.42	32,418.20	31.57	29,676.70	31.54	29,182.60	4.49	30,441.40
100	13.12	56,672.20	77.23	52,260.20	77.14	50,702.90	13.29	52,790.50

Table 3.13. Dispatching Rules, RM2, Exponential Repair Times

Dispatching Rule		
ATC	ProATC1	ProATC2
Objective	Objective	Objective
904.49	1,204.02	902.99
3,115.63	3,906.07	3,129.30
6,382.19	8,546.33	6,389.85
11,301.30	14,081.70	11,248.30
17,412.10	22,562.80	17,160.90
37,634.10	47,567.30	37,413.10
69,155.10	84,180.40	67,956.20

Table 3.14. Comparison of Algorithms, RM2, Exponential Repair Times

Beam Search						Surrogate BS-M1	
Classical BS		Simulation-based BS		Proactive BS			
CPU Time	Objective	CPU Time	Objective	CPU Time	Objective	CPU Time	Objective
0.00	1,033.22	0.05	848.80	0.05	852.23	0.00	1,021.84
0.04	3,163.42	0.46	2,858.13	0.45	2,843.37	0.04	3,056.68
0.15	6,933.54	1.52	6,192.29	1.52	6,126.35	0.15	6,597.78
0.44	11,152.00	3.68	10,092.00	3.68	9,917.35	0.44	10,572.50
0.97	17,693.20	7.27	15,994.40	7.26	15,589.20	0.99	16,530.50
4.43	36,327.90	25.49	33,075.00	25.48	32,261.40	4.55	33,712.50
13.13	63,984.20	62.77	58,651.80	62.76	56,361.90	13.52	58,573.90

Recall that ProATC1 inserts the repair times for all jobs in proportion to their processing times. ProATC2, however, inserts the repair times by estimating the locations of the machine breakdowns in the sequence. Our results indicate that the latter method (ProATC2) performs significantly better than the former approach (ProATC1) and classical ATC.

Our second main observation is that classical ATC is better than the classical BS for 10-, 20-, 30-, and 40-job problems. Beam search yields better performance than ATC only for large problems. On the other hand, however, simulation-based BS is better than all ATC versions for all problem sizes. This indicates that using simulation as a global evaluation function improves the proposed BS significantly. Nevertheless, we should also note that using simulation as a global evaluation function increases the CPU times exponentially with increasing problem sizes.

In the final observation, we note that the advantage of using the proactive approach becomes more significant for large problem sizes. For example, simulation-based BS yields better results for 10-job problems whereas proactive BS is better for 20 or more job problems. Also, ProATC2 displays progressively better performance than ATC when the problem size increases. Similarly, we observe that BS-M1 gets better with increasing problem sizes.

In summary, we can conclude that the proposed beam-search algorithm is quite promising for generating robust or stable schedules. It can also handle

computationally intractable cases such as problems with a general machine breakdown/repair process.

3.9 Concluding Remarks

In this chapter, we study proactive scheduling in a single machine environment with random processing times and random machine breakdowns. We use an expected performance measure as the robustness criterion. We also consider three stability measures. Formal probability theory is used to analyze these measures and some optimality conditions are developed. In this study, we develop an exact algorithm for single machine scheduling problems with processing time uncertainties. We also develop a beam-search algorithm as a heuristic to handle cases with machine breakdown/repair.

Minimizing expected total weighted flow time in a single machine environment subject to random machine breakdowns is known to be $\mathcal{NP}$-hard. Even though the status of the unweighted case (minimizing *RM1*) is unknown, it can be said that the problem is analytically intractable, for it is difficult even to calculate the objective function value of a given solution. For the special case where job processing times are stochastically orderable, Theorem 3.1 gives the optimal solution.

As for *RM2*, Theorem 3.3 gives a dominance rule for the case where no breakdowns are present and job processing times are stochastically orderable. Consideration of breakdowns or relaxing the stochastically orderable assumption quickly renders the problem analytically intractable, for it is known that the problem is $\mathcal{NP}$-hard, as stated in Theorem 3.2.

SM1 and *SM2* are closely related, thus we summarize them together. Sequencing the jobs according to a non-decreasing order of job processing time variances (SVPT) is optimal if no machine breakdowns are present (Theorem 3.4). If machine breakdowns are included, the SVPT rule is still optimal when the uptimes are exponential and the SVPT sequence coincides with the SEPT sequence (Theorem 3.5). Relaxing either of these assumptions, i.e., exponential uptimes or coincidence of

SEPT and SVPT, the problem becomes analytically intractable. Just as in the case of minimizing *RM1*, even the objective function of a given feasible solution cannot be calculated analytically.

If the processing times are not random variables and the machine uptimes are exponentially distributed, SPT is optimal for minimizing *SM3* (Theorem 3.6). Relaxation of either of these assumptions again renders the problem analytically intractable.

To sum up, in this chapter, we model uncertainty regarding job processing times and machine reliability with known probability distributions. We define several robustness and stability measures. This chapter contributes to the existing proactive scheduling literature in two ways: first, we identify the analytically tractable cases and we develop an exact algorithm to solve the common problem of minimizing the expected total tardiness using the insights gained while studying these cases. Second, for intractable cases, rather than taking an indirect approach by employing surrogate measures, we estimate the actual measures directly using simulation. The use of simulation in the existing studies may have been avoided because of its anticipated high computational burden. Our computational results, however, indicate that a beam-search algorithm that employs simulation as a global evaluation function is quite promising and requires reasonable computational times.

We can identify several further research directions. First, the proposed beam-search algorithm can be extended to more general multi-machine environments. Additionally, the job population in this chapter is fixed and all jobs are available at time 0. Inclusion of non-zero ready times and dynamic job arrivals will make the approach more applicable to real-life problems.

Second, robustness can be measured from a different point of view. For example, the notion of a β-robust schedule is used for the total flow time measure in the literature. A β-robust schedule maximizes the probability of achieving a system performance less than or equal to a given level T (Daniels and Carillo, 1997). The same concept can be used when the performance measure is total tardiness. Along the same lines, new, easy-to-calculate robustness or stability measures can be developed. The insight gained from this study suggests that it is hard to find an exact method even when we slightly relax the optimality conditions in the theorems developed in

Sections 3.4 and 3.5 of this chapter. In fact, there are other approaches in the literature that are used when dealing with uncertainty, including scenario planning and modeling with fuzzy numbers. We believe that such approaches could help alleviate the problems encountered in an analytical approach, such as the one taken in this study.

Finally, both robustness and stability are important performance measures for the practitioners. A bicriteria algorithm that can handle both measures is of practical importance. The relationship and the tradeoff between robustness and stability can also be analyzed. This is the topic of next chapter.

Chapter 4

Bicriteria Approach for the Single Machine Environment

4.1 Introduction

In the previous chapter, we have focused on schedule robustness and stability separately. In practice, however, scheduling quality is a multidimensional issue (Kempf, Uzsoy, Smith and Gary, 2000). A decision maker who wants to protect the generated schedule from the negative impacts of uncertainty, for instance, may want to judge the schedule on the basis of both robustness and stability. If only one of them is taken into account, the resulting schedule may be unbalanced. If robustness is the only criterion that is considered, then the resulting schedule may be prone to substantial changes, and therefore to create system nervousness. On the other hand, if the main goal is to optimize stability, then the schedule performance is likely to be poor. These two extremes can be thought of as travelling by motorbike or by lorry, respectively. In order to reach an acceptable compromise, like travelling by automobile, the decision maker has to consider both criteria. Considering robustness and stability simultaneously is the topic of this chapter.

Consideration of multiple criteria in the scheduling studies dates back to 1980s. Since then, a significant volume of literature has emerged. Especially the natural bicriteria model of earliness-tardiness scheduling has been subject to considerable research interest. We refer the interested reader to T'kindt and Billaut (2002) and Hoogeveen (2005) for an elaborate discussion and review of multicriteria scheduling.

There are two general approaches to multicriteria optimization. One is to combine the individual criteria into a single composite criterion. The other is to generate a set of solutions that contains an optimal solution for each reasonable composite criterion that one can think of (the set of Pareto optimal solutions). Evans (1984) identifies these approaches as *a priori* and *a posteriori* optimization, respectively. In this chapter, we consider a posteriori optimization of robustness and stability simultaneously in a single machine environment with processing time variability. We generate the set of all Pareto optimal points via so called ε-constraint method. We formulate the sub-problems required by the method and establish their computational complexity status. A variant of the method that works with only a single type of sub-problem is also considered. A dominance rule and alternative ways to enforce the rule to strengthen the single sub-problem variant of the method are discussed. The performances of the method and the dominance rule are evaluated in an experimental study.

The rest of this chapter is organized as follows: In Section 4.2, we introduce the specific problem that we study in this chapter. Section 4.3 presents the famous ε−constraint method. The computational experiments are explained in Section 4.4. We discuss an approach to reduce the number of generated Pareto optimal points in Section 4.5. We conclude the chapter with a discussion and possible extension in Section 4.6.

4.2 Problem Definition and Notation

In this chapter, we consider the single machine scheduling problem with random processing times. Let $J = \{1, 2, \ldots, n\}$ be the index set of n jobs to be processed on the machine. We assume that all n jobs are released at time $t = 0$. The processing times of the jobs are assumed to be independent random variables with known general cumulative distribution functions that may differ from job to job. Let X_j denote the processing time of job j. We assume $E[X_j] = a_j \in \mathbb{Z}^+ < \infty$ and $Var[X_j] = b_j \in \mathbb{Z}^+ < \infty$ for $j = 1, 2, \ldots, n$. In other words, we have finite, nonnegative integer mean and variance values for all jobs. Given a schedule σ, let $C_j(\sigma)$ denote the completion time

of job *j* in σ after processing times materializes. Note that $C_j(\sigma)$ is a random variable. The robustness and stability measures we use are $R(\sigma) = E[\sum_{i=1}^{n} C_j(\sigma)]$ and $S(\sigma) = \sum_{i=1}^{n} Var[C_j(\sigma)]$, respectively. Note that the value of neither objective function can be made better by inserting idle time into a schedule. Hence, we confine the feasible region to the class of non-delay schedules without loss of generality and a schedule is fully identified by the sequence of its jobs (i.e., by a permutation of *J*). Our objective is to minimize $f(R(\sigma), S(\sigma))$ for all possible composite objective functions *f*, where $f:\Re^2 \rightarrow \Re$ is non-decreasing in both arguments. This assumption on *f* is very natural and reflects the fact that we want to minimize both measures. It additionally has the very convenient effect that under this assumption there exists a *Pareto optimal* or *non-dominated* schedule by which an optimum is attained.

Definition 4.1: *A schedule* σ *is said to be* dominated *by a schedule* π *(denoted by* $\pi \prec \sigma$*) if* $R(\pi) \leq R(\sigma)$ *and* $S(\pi) \leq S(\sigma)$*, where at least one of the inequalities is strict.*

Note that dominance is a transitive relation.

Definition 4.2. *A schedule* σ *is* Pareto optimal *(or* non-dominated*) if there does not exist a schedule* π *such that* $\pi \prec \sigma$.

A special category of dominated schedules is formed by the *weak Pareto optimal* schedules, which are defined as follows:

Definition 4.3. *A feasible schedule* σ *is* weak Pareto optimal *if it is not Pareto optimal but there does not exist a feasible schedule* π *such that* $R(\pi) < R(\sigma)$ *and* $S(\pi) < S(\sigma)$.

Theorem 4.1. *Every non-Pareto schedule is dominated by a Pareto optimal schedule.*

Theorem 4.2. *If* $f:\Re^2 \to \Re$ *is non-decreasing in both arguments, then there exists a Pareto optimal schedule that is in* $\arg\min_{\sigma} f(R(\sigma), S(\sigma))$.

Proofs of Theorem 4.1 and 4.2 are omitted (see e.g., T'kindt and Billaut, 2002). Next, we show that if *f* is linear, an optimum is attained by an *extreme point*.

Definition 4.4. *The cartesian coordinate plot of the points*

$\{(x, y) \in \Re^2 \mid x = R(\sigma), y = S(\sigma), \forall \sigma \text{ such that } \sigma \text{ is Pareto}\}$ *is called* the R-S plot.

Definition 4.5. *Lower envelop of the convex hull of Pareto points in the R-S plot is called* the efficient frontier.

Definition 4.6. *Vertices of the efficient frontier are called* extreme points.

Theorem 4.3. *If the composite objective function f is linear, there exists an extreme point that minimizes f.*

Proof. By Theorem 4.1, it is sufficient to check Pareto points for an optimal solution. Let $\Re$ be the convex hull of Pareto points. Then an optimal point in the R-S plot can be found by solving the following linear programming problem: $\min_{x,y}\{f(x, y) = \alpha x + \beta y \mid (x, y) \in \Re\}$. Fundamental theorem of the linear programming combined with the fact that the steepest descend direction $-\nabla f = (-\alpha, -\beta)^T$ points towards the origin (assuming $\alpha \geq 0$, $\beta \geq 0$) establishes the desired result. □

One needs to search only in the set of Pareto optimal schedules to minimize *any* reasonable composite objective function *f* by Theorem 4.2. Hence, finding the set of Pareto optimal points is a viable idea when dealing with a bicriteria problem where the decision maker wants to minimize both criteria simultaneously, like the one we consider in this chapter. Theorem 4.3 suggests that minimizing a single linear objective function repetitively with various weights may yield only the extreme points. This is can be illustrated with the following example. Consider the jobs whose processing times have the mean and variances values given in Table 4.1. Figure 4.1 depicts the robustness and stability values of all 120 feasible schedules. The Pareto optimal points are marked with a different color in Figure 4.1. By minimizing a linear composite function $\alpha.R(\sigma) + \beta S(\sigma)$, one can obtain only the Pareto points that are marked with squares in the figure; the points denoted by triangles are missed.

In the next section, we present an approach called the *ε-constraint method* (Chankong and Haimes, 1983) that can be used to generate all Pareto points.

4.3 ε–Constraint Method

The following theorem presents the key observation for the ε-constraint method.

Theorem 4.4. *Let* $y^* = \min_{\sigma}\{S(\sigma) | R(\sigma) \le \varepsilon\}$ *and* $x^* = \min_{\sigma}\{R(\sigma) | S(\sigma) \le y^*\}$. *The point* (x^*, y^*) *is Pareto optimal.*

Table 4.1. Numerical Example

	Processing Time	
Job	**Mean**	**Variance**
1	49	91
2	54	58
3	86	24
4	87	1
5	32	61

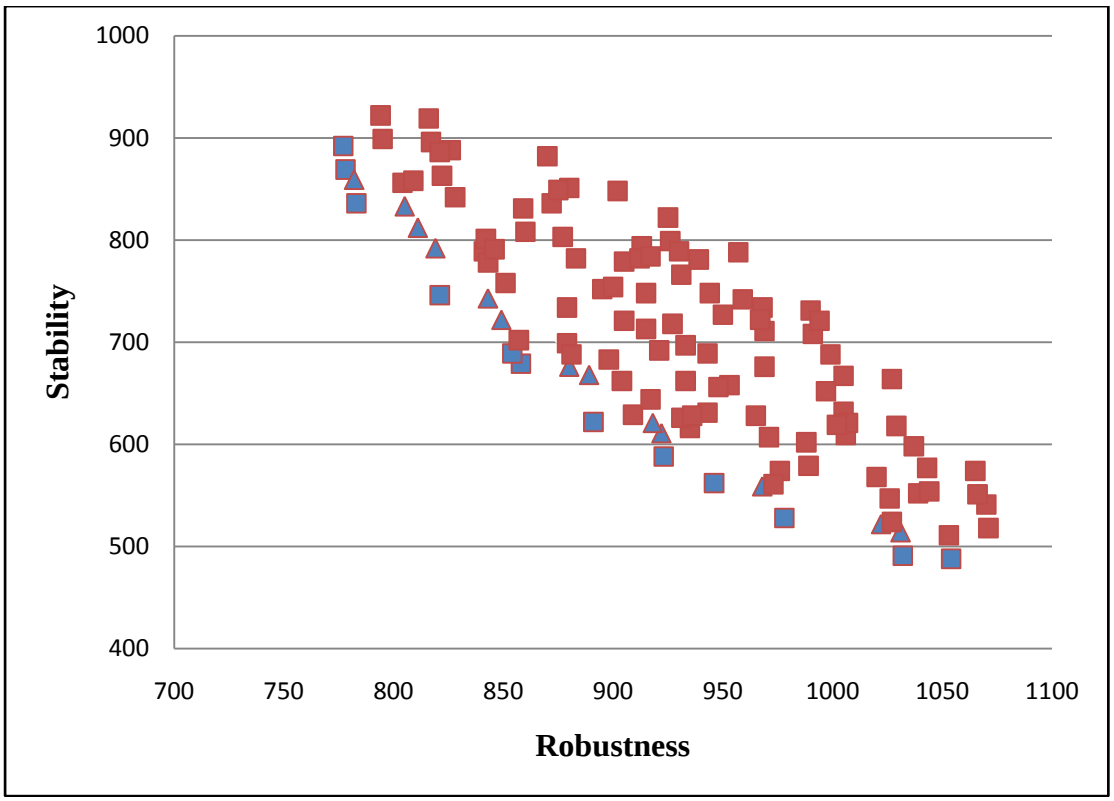


Figure 4.1. Numerical Example

For the proof of Theorem 4.4 with any two criteria (which may be different than R(σ) and S(σ)), see T'kindt and Billaut (2002).

Let $R^{LB} = \min_{\sigma}\{R(\sigma)\}$, $S^{LB} = \min_{\sigma}\{S(\sigma)\}$, $R^{UB} = \min_{\sigma \in \arg\min S(\sigma)}\{R(\sigma)\}$, and $S^{UB} = \min_{\sigma \in \arg\min R(\sigma)}\{S(\sigma)\}$.

Theorem 4.4 can be used to generate the set of Pareto optima as in the following algorithm.

ALGORITHM ε-CONSTRAINT1:

Step 1.

k ← 1.

Begin with the first Pareto optimal point (x_0, y_0) where $x_0 = R^{UB}$ and $y_0 = S^{LB}$.

Step 2.

Obtain the next Pareto optimal point (x_k, y_k) where $y_k = \min\{S(\sigma) \mid R(\sigma) \le x_{k-1} - 1\}$ and $x_k = \min\{R(\sigma) \mid S(\sigma) \le y_k\}$.

Step 3.

If (x_k, y_k) = (R^{LB}, S^{UB}) STOP.

Else $k \leftarrow k + 1$.

Goto Step 2.

Note that in Step 2, had we not integer mean and variance values, the correct problem to solve for finding the *y* value that corresponds to the next Pareto point would be $y_k = \min\{S(\sigma) \mid R(\sigma) < x_{k-1}\}$.

The following 0-1 integer programming formulations can be used to solve the two minimization problems in Step 2.

Let $z_{ji} = \begin{cases} 1, & \text{if job } j \text{ is in position } i \text{ in the schedule} \\ 0, & \text{otherwise} \end{cases}$

Problem P1:	Problem P2:
$y_k = \min \sum_{i=1}^{n}\sum_{j=1}^{n}(n-i+1)b_j z_{ji}$	$x_k = \min \sum_{i=1}^{n}\sum_{j=1}^{n}(n-i+1)a_j z_{ji}$
subject to	subject to
$\sum_{i=1}^{n} z_{ji} = 1, \quad \forall j$	$\sum_{i=1}^{n} z_{ji} = 1, \quad \forall j$
$\sum_{j=1}^{n} z_{ji} = 1, \quad \forall i$	$\sum_{j=1}^{n} z_{ji} = 1, \quad \forall i$
$\sum_{i=1}^{n}\sum_{j=1}^{n}(n-i+1)a_j z_{ji} \le x_{k-1} - 1,$	$\sum_{i=1}^{n}\sum_{j=1}^{n}(n-i+1)b_j z_{ji} \le y_k,$
$z_{ji} \in \{0,1\} \quad \forall j, \forall i$	$z_{ji} \in \{0,1\} \quad \forall j, \forall i$

P1 is a binary IP formulation for $\min\{S(\sigma) \mid R(\sigma) \le x_{k-1} - 1\}$ and P2 is the same for $\min\{R(\sigma) \mid S(\sigma) \le y_k\}$. Consider the objective functions in these formulations. For any position *i* in the schedule, only one of the terms of the inner summation is positive

(z_{ji} = 1) and the rest are all zero (z_{ji} = 0). The positive term corresponds to a job *j* whose contribution to the objective function value is (n - i + 1) times variance (b_j) or mean (a_j) of the processing time of that job. Note that this contribution is equivalent to the variance or mean of the completion time of that job. Summation of this over all positions (*i*) yields the stability and the robustness measures, respectively. As for constraints, the first two sets are assignment constraints and establish a one-to-one relation between jobs and positions. The third set of constraints (which is a singleton) places an upper bound on the secondary (i.e., the one that is not being minimized) measure.

The unconstrained versions (i.e., without the third set of constraints) of P1 and P2 that are needed in Step 1 and Step 3 can be solved in O(n log n) time by the Shortest Expected Processing Time (SEPT) first and Shortest Variance of Processing Time (SVPT) first rules, respectively. The constrained versions in Step 2 are 0-1 assignment problems with single side constraints, which are known to be $\mathcal{NP}$-hard (Mazzola and Neebe, 1986).

Problems P1 and P2 can be solved using the following forward dynamic programming formulations. Let *J* be a subset of the *n* jobs and assume that the jobs in *J* are processed first.

Problem P1:

Let

$$V(J,r)=\begin{cases}\sum_{j\in J} Var\left[C_j\right], & \text{if } \sum_{j\in J} E\left[C_j\right]\le r\\ \infty, & \text{otherwise}\end{cases}.$$

Initial Conditions:

$$V(\{j\},r)=\begin{cases}Var\left[C_j\right]=b_j, & \text{if } E\left[C_j\right]=a_j\le r\\ \infty, & \text{otherwise}\end{cases},\quad j=1,\cdots,n$$

Recursive Relation:

$$V(J,r)=\min_{j\in J}\left\{V\left(J-\{j\},r-(n-|J|+1)\cdot a_j\right)+(n-|J|+1)\cdot b_j\right\}$$

Optimal Value Function:

$$V(\{1,\cdots,n\}, x_{k-1}-1)$$

Problem P2:

Let

$$V(J,s)=\begin{cases}\sum_{j\in J} E[C_j], & \text{if } \sum_{j\in J} Var[C_j]\le s\\ \infty, & \text{otherwise}\end{cases}.$$

Initial Conditions:

$$V(\{j\},s)=\begin{cases}E[C_j]=a_j, & \text{if } Var[C_j]=b_j\le s\\ \infty, & \text{otherwise}\end{cases},\quad j=1,\cdots,n$$

Recursive Relation:

$$V(J,s)=\min_{j\in J}\{V(J-\{j\}, s-(n-|J|+1)\cdot b_j)+(n-|J|+1)\cdot a_j\}$$

Optimal Value Function:

$$V(\{1,\cdots,n\}, y_k)$$

The idea behind these formulations is relatively straightforward. An optimal sequence for a subset of the jobs is determined at each iteration, assuming this subset is scheduled first. This is done for every subset of a constant cardinality *c*. The contribution of the *c* scheduled jobs to the objective is calculated for each subset. Recursive relation is used to expand the considered subsets by one job to obtain the subsets of cardinality *c* + 1. Each job in the expanded subset is considered to be the last. When using the recursive relation, the actual sequence of the *c* jobs in the smaller subset is not required; only the contribution to the objective has to be known. After determining $V(\{1,\cdots,n\},.)$, an optimal sequence can be obtained with a simple backtrack. Since there are a total of 2^n subsets of {1, …, n}, the time complexity for the above dynamic programming formulations are $O((x_{k-1} - 1).2^n)$ and $O(y_k.2n)$ for P1 and P2, respectively. Although these values can be an improvement over the brute

force method of total enumeration (O(n!)), the amount of required space to store the computed V(J, .) values is too prohibitive to solve large problems.

Theorem 4.5. *Let x_0, y_0 be a Pareto point. Let* $\sigma_0 \in \arg\min\{S(\sigma) \mid R(\sigma) \le x_0 - 1\}$. *Then $(x_1, y_1) = (R(\sigma_0), S(\sigma_0))$ is either a Pareto point or a weak Pareto point.*

Theorem 4.5, which we present without proof (see T'kindt and Billaut, 2002), makes it possible to solve only Problem P1 instances to generate all Pareto points as demonstrated in the following algorithm listing.

ALGORITHM ε-CONSTRAINT2:

Step 1.

$k \leftarrow 1$.

Begin with the first Pareto optimal point (x_0, y_0) where $x_0 = R^{UB}$ and $y_0 = S^{LB}$.

Step 2.

Obtain the next Pareto candidate $(x_k, y_k) = (R(\sigma_k), S(\sigma_k))$ where $\sigma_k \in \min\{S(\sigma) \mid R(\sigma) \le x_{k-1} - 1\}$.

If ($y_k = y_{k-1}$), previously obtained candidate (x_{k-1}, y_{k-1}) is not Pareto optimal. Eliminate it.

Step 3.

If $(x_k, y_k) = (R^{LB}, S^{UB})$ STOP.

Else $k \leftarrow k + 1$.

Goto Step 2.

Recall that ε-CONSTRAINT1 solves a Problem P1 instance and a Problem P2 instance to obtain the next Pareto optimal point. On the other hand, ε-CONSTRAINT2 solves only a single instance of Problem P1 to obtain the next point. This point is either a Pareto or a weak Pareto optimal point by Theorem 4.5. The drawback of the latter is that newly generated points are not guaranteed to be Pareto optimal. Eventually the whole set of Pareto optimal points is generated by

removing the weak Pareto optimal points, but the number of iterations required may be more than the former algorithm.

Some of the weak Pareto optimal points may be avoided with the help of the dominance rule that is established in the following theorem.

Theorem 4.6. *Consider a job pair (u, v). If $a_u \leq a_v$ and $b_u \leq b_v$,with at least one of the inequalities is strict, job u precedes job v in any Pareto optimal schedule.*

Proof. Assume σ is a Pareto optimal schedule in which job *v* precedes job *u*. Say jobs *v* and *u* occupy p^{th} and r^{th} ($p < r$) positions, respectively, in σ. Obtain a new schedule σ′ by swapping the positions of jobs *u* and *v*. In other words, in σ′, jobs *v* and *u* occupy r^{th} and p^{th} positions, respectively, and the positions of other jobs are not changed. The contributions of jobs other than *u* and *v* to robustness and stability measures are the same in both schedules. We have

$R(\sigma) - R(\sigma') = (n - p + 1)a_v + (n - r + 1)a_u - (n - r + 1)a_v - (n - p + 1)a_u$

$= (r - p)(a_v - a_u) \geq 0.$

Similarly,

$S(\sigma) - S(\sigma') = (n - p + 1)b_v + (n - r + 1)b_u - (n - r + 1)b_v - (n - p + 1)b_u$

$= (r - p)(b_v - b_u) \geq 0.$

In other words, we have $R(\sigma') \leq R(\sigma)$ and $S(\sigma') \leq S(\sigma)$. Moreover, since at least one of $(a_v - a_u)$ and $(b_v - b_u)$ is strictly positive, at least one of $R(\sigma') \leq R(\sigma)$ and $S(\sigma') \leq S(\sigma)$ is strict. In other words, σ is dominated by σ′ which contradicts with the assumption that σ is Pareto optimal. □

All job pairs that satisfy the condition stated in the hypothesis of Theorem 4.6 can be identified in $O(n^2)$ time. For each such (*u*, *v*) pair, Table 4.2 presents three

Table 4.2. Enforcing “u precedes v”

Constraint Set	**Number of Additional Rows**	**Total Number of Nonzeros**
$\sum_{k=1}^{i} z_{uk} \geq \sum_{k=1}^{i} z_{vk}, \quad \forall i$	n	n(n+1)
$z_{vi} \leq \sum_{k=1}^{i-1} z_{uk}, \quad \forall i \geq 2$	n-1	n(n+1)/2 - 1
$\sum_{i=1}^{n} i z_{vi} - \sum_{i=1}^{n} i z_{ui} \geq 1$	1	2n

different constraint sets that can be appended to the previously given formulation of Problem P1 to enforce the dominance rule.

The first summation in the first constraint set acts as an indicator of whether or not job *u* is placed in the first *i* positions. The second summation is the same for job *v*. In other words, the first constraint set ensures that if job *u* is not among the first *i* jobs of the schedule, then neither is job *v*. Similarly, the second constraint set ensures that if job *v* is the job that is in position *i*, then job *u* is should be among the first *i - 1* jobs of the schedule. The first summation in the third constraint set (in fact it is only a single constraint) is the position in which job *v* is scheduled. The second summation is the same for job *u*. Hence, the third constraint set ensures that the difference between the positions in which jobs *v* and *u* are scheduled is at least 1, with job *v* having the later position.

4.4 Computational Experiments

The quality of the proposed ε-constraint method (the second variation) and the performance of the alternative formulations of the dominance rule are assessed on several input problems. We conjecture that the total number of Pareto optimal points is pseudo polynomial and depends on the processing time mean and variance values. To investigate the validity of this conjecture, a computation test bed is prepared to include two levels of processing time means. The means are sampled from the discrete

uniform distributions U(0, 50) and U(0, 100). It is expected that more Pareto optimal points will be observed in the latter level. The effect of correlation between processing time means and variances is also examined. If there is a positive correlation, the two criteria that are considered, namely robustness and stability, are expected to be less conflicting whereas if there is a negative correlation, minimizing one would probably worsen the other criterion. Hence, in the case of a positive correlation, the number of Pareto optimal points is expected to be less. Three levels of correlation are considered in our computational experiments: negative, none and positive. To induce a negative correlation between processing time means and variances, first, the generated value of the mean is checked: if it is less than the expectation (μ) of its respective distribution, the variance is sampled from the discrete uniform distribution U(μ, 2μ); otherwise it is sampled from discrete U(0, μ). Similarly, to induce a positive correlation, if the generated value of the mean is less than μ, the variance is sampled from discrete U(0, μ), otherwise it is sampled from discrete U(μ , 2μ). For the uncorrelated case, the variance is sampled from discrete U(0, 2μ), without inspecting on the actual value of the mean. Finally, three levels of problem size is considered: 10-, 30- and 50-job problems. To sum up, the experimental design consists of 18 problem classes (3 levels of size x 3 levels of correlation x 2 levels of mean range). For each problem class, 10 instances are generated, resulting in a total of 180 problem instances.

The ε-constraint method is coded in the C++ language and run on a Linux box running CentOS on a dual core AMD Opteron 252 – 2.6GHz system with 2GBs of physical RAM. The constrained version of Problem P1 is repeatedly solved utilizing ILOG CPLEX 8.1 callable library.

4.4.1 Different Formulations for the Dominance Rule

All three versions of constraint sets presented in Table 4.2 along with using no dominance rule are compared on 10-job problems. The number of weak Pareto optimal points are generated is given in Table 4.3. Table 4.4 presents the CPU seconds used by the algorithm. The numbers in Tables 4.3 and 4.4 are the averages for the 10 instances of the corresponding problem classes.

Table 4.3. Number of Weak Pareto Optimal Points with Different Dominance Rule Formulations

	No Dominance Rule			**Constraint Set 1**		
	Correlation Level			**Correlation Level**		
UB on Mean	**N**	**O**	**P**	**N**	**O**	**P**
50	7.8	3.2	0.3	7.1	3.8	0.5
100	10.2	2.5	1.3	7.4	2.6	1.3
	Constrain Set 2			**Constraint Set 3**		
	Correlation Level			**Correlation Level**		
UB on Mean	**N**	**O**	**P**	**N**	**O**	**P**
50	7.7	3.4	0.3	6.8	3.4	0.4
100	7.0	2.9	1.7	7.0	2.1	1.3

Table 4.4. CPU seconds with Different Dominance Rule Formulations

	No Dominance Rule			**Constraint Set 1**		
	Correlation Level			**Correlation Level**		
UB on Mean	**N**	**O**	**P**	**N**	**O**	**P**
50	0.983	0.221	0.044	1.888	0.783	0.260
100	1.774	0.345	0.121	3.049	1.016	0.483
	Constrain Set 2			**Constraint Set 3**		
	Correlation Level			**Correlation Level**		
UB on Mean	**N**	**O**	**P**	**N**	**O**	**P**
50	1.738	0.622	0.180	1.144	0.330	0.085
100	2.946	0.855	0.392	2.034	0.483	0.196

On examining Tables 4.3 and 4.4 we observe that as the number of rows and nonzero coefficients appended to the problem decrease, the required CPU seconds also decrease as expected. In terms of CPU seconds, formulation 3 performs the best. Note that adding constraints to enforce the dominance rule may lead into an increase in the number of weak Pareto optimal points visited. This somewhat counterintuitive observation can be explained as follows. Let set S consist of the schedules with the same particular value of the stability measure, say y_k. If y_k happens to be the optimal objective function value for the problem P1 at the k^{th} iteration of the ε-constraint method, let S be the set of alternative optima for P1 at that iteration. Consider the

subset S' of S that consists of the schedules whose value of the robustness measure is less than x_{k-1}. All schedules in S', except for one, are weak Pareto optimal. Consider the subset S'' of the set S' which consists of the schedules that comply with the dominance rules that is stated in Theorem 4.6. We know that S'' contains the Pareto optimal point that we are after, but it is not guaranteed that it will be selected among the alternative optima. The dominance constraints only ensure that the selected schedule will be in S'' instead of the larger set S'. It is, however, perfectly possible for CPLEX, on solving a model with some dominance constraints, to pick a schedule σ in S'' with a greater $R(\sigma)$ value compared with the solution of a model without any dominance constraints. Note that dominance rule performs better in the case of a negative correlation, under which minimizing stability would probably worsen robustness and create room for the dominance rule to rectify the robustness value. On the other hand, in the case of a positive correlation, the act of minimizing the stability measure would probably also inherently minimize the robustness measure and the "unnecessary interference" of the dominance constraints would do more harm than good.

Although constraint set 3 performs the best in terms of both, the number of weak Pareto points and CPU seconds, we exclude the dominance constraints from the rest of our experimentation since generating weak Pareto points and eliminating them is less costly than avoiding them through the dominance rule in terms of CPU seconds.

4.4.2 Effect of the Problem Size, Correlation and Mean Range

The rest of the experimentation is carried out without any type of dominance constraints. Tables 4.5 - 4.8 give the number of Pareto/weak Pareto optimal points when the processing time means are sampled from U(0, 50) and U(0, 100). Tables 4.9 and 4.10 present the CPU seconds. The numbers in the tables are the averages of 10 problem instances in the corresponding problem class. Figure 4.2 displays the data from Tables 4.5 - 4.8 in a more visual fashion.

Table 4.5. Number of Pareto Optimal Points for $X_i \sim U(0, 50)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	171.4	57.9	18.7
30	4588.4	2047	507.4
50	13398.7	6145.6	1417.6

Table 4.6. Number of Weak Pareto Optimal Points for $X_i \sim U(0, 50)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	7.8	3.2	0.3
30	496.2	320.8	53.2
50	1277.9	887.9	144.6

Table 4.7. Number of Pareto Optimal Points for $X_i \sim U(0, 100)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	215.3	73.9	32.1
30	7882.2	2918.8	980.7
50	25217.4	11402.8	3083.7

Table 4.8. Number of Weak Pareto Optimal Points for $X_i \sim U(0, 100)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	10.2	2.5	1.3
30	1201.8	454.9	138.1
50	3695.9	2022.8	489.4

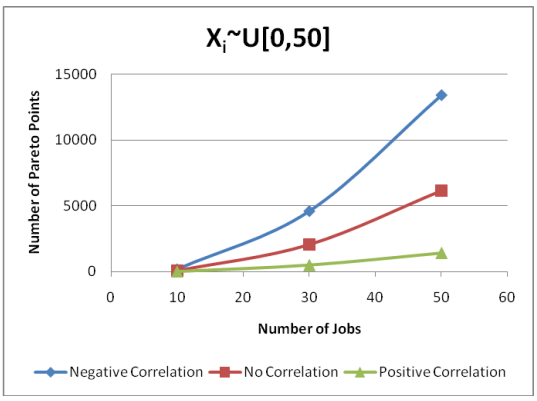
Table 4.9. CPU Seconds for $X_i \sim U(0, 50)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	0.98	0.22	0.04
30	203.22	83.53	13.63
50	2128.12	853.86	126.34

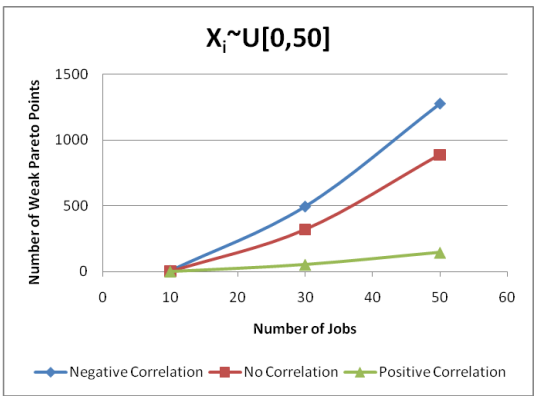
Table 4.10. CPU Seconds for $X_i \sim U(0, 100)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	1.77	0.35	0.12
30	501.60	148.53	33.20
50	4870.44	1838.50	382.24

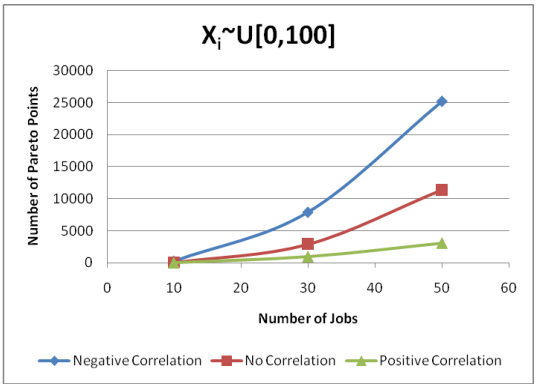
On examining the tables and the figure we observe that number of Pareto optimal points increase with 1) increasing levels of problem size, 2) decreasing levels of correlation (from positive to nil, then to negative) between the means and the variances of processing times and 3) increasing levels of processing time mean range. All three observations are intuitive and were expected before the computational experiments.



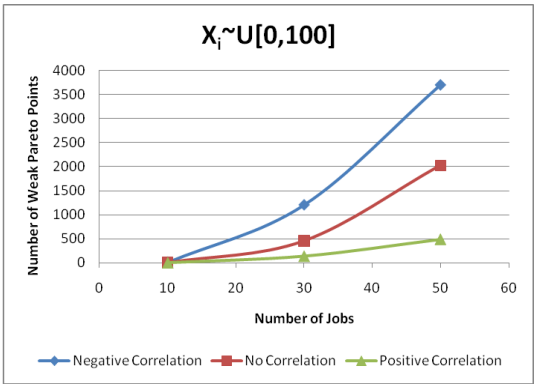
a. Number of Pareto Points for $X_i \sim U(0, 50)$



b. Number of Pareto Points for $X_i \sim U(0, 100)$



c. Number of Weak Pareto Points for $X_i \sim U(0, 50)$



d. Number of Weak Pareto Points for $X_i \sim U(0, 100)$

Figure 4.2. Number of Pareto and Weak Pareto Points

These results also provide evidence to the conjecture that the number of Pareto points depend on the processing time mean and variance values and may be pseudo polynomial in number of jobs, bounded from above with $\min\{R^{UB} - R^{LB} + 1, S^{UB} - S^{LB} + 1\}$.

For all problem classes, the weak Pareto points constitute a small percentage of the whole set of candidate points and their number is far away from being close to the number of Pareto points. This justifies the use of ε-CONSTRAINT2 variant instead of ε-CONSTRAINT1, which solves twice as many problems to generate a single point.

ε-CONSTRAINT1 would be beneficial if the number of the weak Pareto points were more than the number of Pareto points.

Another possible way to avoid weak Pareto points is shown by the following theorem.

Theorem 4.7. *Let x_0, y_0 be a Pareto point. Let*

$\sigma_0 \in \arg\min_{\sigma} \{S(\sigma) + \alpha \cdot R(\sigma) \mid R(\sigma) \le x_0 - 1\}$ *where* $\alpha = 1 / (R^{UB} - R^{LB} + 1)$. *Then* $(x_1, y_1) = (R(\sigma_0), S(\sigma_0))$ *is a Pareto point.*

Theorem 4.7 enables us to use the following ε-constraint version where only a single problem is solved and the generated points are guaranteed to be Pareto optimal.

ALGORITHM ε-CONSTRAINT3:

Step 1.

$k \leftarrow 1$.

Begin with the first Pareto optimal point (x_0, y_0) where $x_0 = R^{UB}$ and $y_0 = S^{LB}$.

Step 2.

Obtain the next Pareto optimal point (x_k, y_k) where $x_k = R(\sigma')$, $y_k = S(\sigma')$, $\sigma' \in \arg\min_{\sigma} \{S(\sigma) + \alpha \cdot R(\sigma) \mid R(\sigma) \le x_{k-1} - 1\}$, $\alpha = 1 / (R^{UB} - R^{LB} + 1)$.

Step 3.

If $(x_k, y_k) = (R^{LB}, S^{UB})$ STOP.

Else $k \leftarrow k + 1$.

Goto Step 2.

See Özlen and Azizoğlu (2009) for a discussion of Theorem 4.7 and algorithm ε-CONSTRAINT3 in the context of bi-objective integer programming problems.

4.5 δ–Grid Search

As explained in the previous section, our computational experiments indicate that the total number of Pareto points increases as the number of jobs increases. Additionally, an increase in mean and variance ranges also causes a rapid increase in total number of points, which gives evidence to support our conjecture about the number of Pareto points being pseudo polynomial in number of jobs. We note that although a single iteration of the ε-CONSTRAINT2 algorithm takes very little computational time, increasing problem sizes cause a rapid increase in total number of Pareto points and hence in total number of iterations required to generate the whole Pareto set. This suggests that being able to define the characteristics and shape of the trade-off curve using fewer Pareto points is of the essence. After all, overwhelming the decision maker with 25,000 Pareto optimal alternatives is not a practical approach. In order to generate a representative subset of the set of Pareto points, we propose an approach which we call the *δ-grid method*. In this method, we use an indifference (or significance) parameter δ.

Definition 4.7. *Let $\delta = (\delta_r, \delta_s) \geq \mathbf{0}$ be the given indifference parameters. Let σ_1 and σ_2 be two feasible schedules. If $|R(\sigma_1) - R(\sigma_2)| < \delta_r$ ($|S(\sigma_1) - S(\sigma_2)| < \delta_s$), the schedules are called δ_r robustness indifferent (δ_s stability indifferent). If $R(\sigma_1) \leq R(\sigma_2) - \delta_r$ and $S(\sigma_1) \leq S(\sigma_2) - \delta_s$, σ_1 is said to δ-dominate σ_2.*

Definition 4.8. *Let $\delta = (\delta_r, \delta_s)$ be the given indifference parameters. A schedule σ is said to be δ-Pareto optimal if there does not exists a feasible schedule π such that π δ-dominates σ.*

If given two schedules σ_1 and σ_2 are δ-robustness (δ-stability) indifferent, the two solutions are regarded to have the same robustness (stability) performance for all practical purposes. To the contrary, if the absolute difference between the values of their robustness (stability) measures is greater than or equal to δ_r (δ_s), these solutions are *significantly different*. For small enough δ (e.g., $\delta < 1$ for integer means or variances of processing time), all feasible solutions are significantly different from one another. In this case each δ-Pareto optimal solution corresponds to a Pareto optimal point. On the other hand, if δ is large enough (e.g., infinity), all solutions become indifferent and the whole Pareto set can be represented by a single δ-Pareto point. In between two extremes, different δ values lead to a different number of δ-Pareto points and δ-grid search aims to find enough number of δ–Pareto points that adequately represents the whole Pareto set. The following algorithm can be used to determine a set of δ–Pareto optimal points.

ALGORITHM δ-GRID:

Step 1.

$k \leftarrow 1$.

Begin with the first Pareto optimal point (x_0, y_0) where $x_0 = R^{UB}$ and $y_0 = S^{LB}$.

Step 2.

Obtain the next Pareto candidate $(x_k, y_k) = (R(\sigma_k), S(\sigma_k))$ where $\sigma_k \in \min\{S(\sigma) \mid R(\sigma) \le G(x_{k-1}) - 1\}$, $G(x) = x_0 + \left\lfloor \frac{x - x_0}{\delta_r} \right\rfloor \cdot \delta_r$.

Step 3.

If $(x_k, y_k) = (R^{LB}, S^{UB})$ STOP.

Else $k \leftarrow k + 1$.

Goto Step 2.

Indifference in the above sense can be geometrically viewed as dividing the *R*-*S* plane into rectangles of size δ_r by δ_s (called *grids*) and treat the solutions that lie within the boundaries of the same grid as one and the same. With this approach, the decision maker is not overwhelmed by thousands of Pareto points. He/she can set the desired granularity level by selecting a particular value for δ, and the resulting set of δ-Pareto points can be seen as alternative optimal schedules or can be used to make inferences about the trade-off between robustness and stability. Note that selecting small enough a δ value leads a set of δ-Pareto points that is equivalent to the set of all Pareto points. Thus, ε-constrained method can be seen as a special case of δ-grid search.

Note that ALGORITHM δ-GRID can generate weak Pareto optimal points. Note also that we do not eliminate the first one of two successive schedules with the same stability value. In other words, the resulting set from ALGORITHM δ-GRID can also contain δ-weak Pareto points (i.e., not all points have to be δ Pareto optimal). We keep the mentioned weak Pareto solutions to keep a uniform spread of generated points. Furthermore, if a dominating Pareto point is within the same grid, the retained weak Pareto point and the dominating point are δ-indifferent so their performances are practically the same. Else, if a dominating Pareto point is not within the same grid (but in a grid with lower robustness limits) the forthcoming iterations of the algorithm will either find that Pareto point or a δ-indifferent weak Pareto point. For we keep weak Pareto points deliberately, we say that ALGORITHM δ-GRID generates *near-Pareto optimal* points.

Selection of the δ value is important in δ-grid approach. Smaller δ values provide a better granularity. The number of generated points, however, could be large. If δ is chosen too large, on the other hand, the number of generated points may be small but the grids could be too large to be of any practical value. In other words, two schedules whose performances are significantly different in practice can be δ-indifferent for large δ values. Unfortunately, there is no *optimal* way to determine which δ value to use.

It is reasonable for the decision maker to determine an upper bound on the total number of generated points. If he/she decides to generate at most N points, the corresponding indifference grid size will be $\delta_r = (R^{UB}–R^{LB}) / N$ by $\delta_s = (S^{UB}–S^{LB}) / N$.

In this chapter, we select to generate at most 1000 near-Pareto points, hence the grids are of $\delta_r = (R^{UB}–R^{LB}) / 1000$ by $\delta_s = (S^{UB}–S^{LB}) / 1000$.

The generated number of points and required CPU seconds are presented in Tables 4.11, 4.12 and 4.13, 4.14, respectively.

Table 4.11. Number of Points for $X_i \sim U(0, 50)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	168.6	59.4	18.3
30	987.2	871	509.8
50	1001.5	968	844

Table 4.12. Number of Points for $X_i \sim U(0, 100)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	212.5	75.3	32.9
30	997.1	913.6	730.2
50	1015.9	982.9	931.6

Table 4.13. CPU Seconds for $X_i \sim U(0, 50)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	0.913	0.205	0.04
30	36.575	27.328	12.34
50	134.938	104.652	65.109

Table 4.14. CPU Seconds for $X_i \sim U(0, 100)$

	Correlation Level		
# of Jobs	**N**	**O**	**P**
10	1.589	0.328	0.117
30	48.871	33.355	20.396
50	157.703	116.463	90.978

In contrast with the exponential increase in the number of Pareto points and the required CPU seconds as the problem size increases, δ-grid search results demonstrate modest increase in both, the number of points and the required computational burden.

4.6 Discussion

In this chapter, we study proactive scheduling in a single machine environment with random processing times. We use total expected flowtime and total variance of job completion times as the robustness and stability measures, respectively. A bicriteria approach to minimize both measures simultaneously is discussed. The proposed ε-constraint method, which generates the set of all Pareto optimal points, is more thorough than the common approach of combining both objective functions into a linear composite objective function. It is frequently used in multi criteria decision making studies in different fields, including machine scheduling. Two different versions of the ε-constraint method is investigated: the first one solves two instances of $\mathcal{NP}$-hard problems to obtain a Pareto optimal point whereas the second one solves only one such problem but the obtained point may be weak Pareto optimal. A dominance rule and three ways to formulate this rule are developed to get rid of some of weak Pareto points in the second variant.

Our computational experiments indicate that incorporating the dominance rule to the problem formulation at each iteration may in fact lower the number of weak Pareto points, especially in the presence of a negative correlation between the processing time mean and variance values. Our experiments, however, demonstrate that generating weak Pareto points and eliminating them is cheaper in terms of computational time than avoiding them. The computational results also show that the presence of a negative correlation between processing time means and variances increase the total number of Pareto optimal points. Total number of Pareto points also increase as the number of jobs increase. Additionally, an increase in mean and variance ranges also causes a rapid increase in total number of points, which gives evidence to our suspect that the number of Pareto points may be pseudo polynomial in number of jobs. We also note that although a single iteration of the algorithm takes very little computational time, increasing problem sizes cause a rapid increase in total number of Pareto points and hence in total number of iterations required to generate the whole set. This suggests that being able to define the characteristics and shape of the trade-off curve using fewer Pareto points is of the essence. To that end, we propose the δ-grid search approach which generates a fixed number (set by the decision maker) of near-Pareto points.

We identify several further research directions. First, the proposed approaches can be extended to other robustness and stability measures. Although multi criteria scheduling is not a new topic, most of the research effort is focused on earliness/tardiness problems or minimizing two regular performance measures at the same time. We believe that using the available toolbox of multi criteria techniques may help decision makers a great deal when coping with uncertainty. Second, the analysis can be extended to the more general shop floor environments such as shops with parallel machines, flow shops or job shops. Finally, algorithms that discover the characteristics of the trade off curve more cleverly may be developed. The brute force approach of generating the whole set of Pareto points may be impractical in terms of computational time requirements. Evolutionary meta heuristics are successfully being used in multicriteria decision making literature for this purpose.

Chapter 5

Job Shop Environment

5.1 Introduction

This chapter can be seen as an extension to the study in Chapter 3. There, we have studied the problem in the single machine environment. In this chapter, we focus on generating stable schedules in a job shop environment with random processing times and machine breakdowns.

Unlike the previous studies in the literature, in this chapter, stability is the primary objective function to optimize. The operation processing times are taken as random variables as well as machine up and down times. The stability measure used in this chapter is the sum of the variances of the realized completion times (*SM*). The problem of minimizing *SM* in a job shop environment subject to random machine breakdowns and processing time variability is called Problem Π. In Section 5.4, we prove that Π is not in the class $\mathcal{NP}$. Hence, a surrogate stability measure (*SSM*) is developed and this version of the problem is called Π'. The problem of minimizing *SSM* in a job shop environment subject to processing time variability only (i.e., no machine breakdowns) is called the problem Π''. In Section 5.4, we prove that Π'' (and therefore Π') is $\mathcal{NP}$-hard. Two exact solution procedures (branch-and-bound algorithms) are developed for the problem Π''. Two heuristics (a beam-search and a tabu-search algorithm) are also developed to handle large instances of Π''. As shown later, calculation of even the surrogate measure (*SSM*) is not possible for Π' in the presence of random machine breakdowns. Thus, the proposed beam-search and tabu-search algorithms are modified in Section 5.8.2 to handle breakdowns. The same

modifications cannot be applied to the branch-and-bound algorithms due to the following two reasons: first, they would lose the property of being exact solution procedures and they are too computationally expensive to use as heuristics, and second, the proposed tabu-search algorithm already performs significantly well, even better than the branch-and-bound algorithms, as shown in Section 5.8.1.2. These problem versions and related solution methods are summarized in Table 5.1.

Table 5.1. Problem Versions

Problem	**Computational Complexity**	**Proposed Solution Methods**
Π	unknown; not in $\mathcal{NP}$	-
Π'	$\mathcal{NP}$-hard	2 branch-and-bound algorithms beam-search tabu-search
Π''	$\mathcal{NP}$-hard	beam-search tabu-search

This chapter extends the stability scheduling literature in four ways: first, a new practical stability measure is defined; second, complexity status of the problems are determined; third, processing time variability and machine breakdowns are simultaneously considered in the problem settings; and finally, two exact solution procedures and heuristics are proposed to solve the problems.

The rest of this chapter is organized as follows. In Section 5.2, we define the problem and introduce the preliminary notation. In Section 5.3, we review the disjunctive graph model for the job shop scheduling problem. We present the stability measure used and the status of the problem in Section 5.4. In Section 5.5, we propose two branch-and-bound algorithms to optimally solve the problem with no breakdowns. Section 5.6 presents a beam-search algorithm which can handle breakdowns and large problems. In Section 5.7, a tabu-search algorithm is developed to further improve the solution quality. Section 5.8 is dedicated to the computational experiments and the presentation of the results. Finally, we make concluding remarks and discuss future research directions in Section 5.9.

5.2 Notation and Problem Definition

Consider the job shop scheduling problem with n jobs and m machines. Each job consists of at most m operations. Each of the operations associated with a job must be carried out in sequence and each operation is associated with a machine. The operation associated with job j and machine i is called operation (i, j). The processing time of operation (i, j) is denoted by a random variable X_{ij} with a general cumulative distribution function $H_{ij}(t)$. Let $a_{ij} = E[X_{ij}]$ and $b_{ij} = Var[X_{ij}]$, where E and V are the expectation and variance operators, respectively. The machines are subject to random breakdowns. The up times for machine i have independent and identical general distribution $G_{i1}(t)$. Similarly, the "down" times (i.e., the times that the machine is not in operation due to breakdown) are independent and identically distributed according to a general distribution $G_{i2}(t)$. Let U_{i1}, U_{i2} … be the sequence of up times and D_{i1}, D_{i2}… be the sequence of down times for machine i. That is, the machine is operational from time 0 until U_{i1}, when the first breakdown occurs. The machine then takes time D_{i1} to be repaired and is again available for processing from time $U_{i1} + D_{i1}$ until time $U_{i1} + D_{i1} + U_{i2}$, and so on. Let C_j denote the time that job j completes its last operation. We assume that all n jobs are ready at time $t = 0$. We denote this stochastic job shop scheduling problem as $Jm \mid X_{ij} \sim H_{ij}(t)$; *brkdwn:* $U_i \sim G_{i1}(t)$, $D_i \sim G_{i2}(t)$; $\beta \mid \gamma$ where $Jm \mid \beta \mid \gamma$ denotes the deterministic counterpart. Here, β denotes the scheduling attributes such as release dates, setup times, preemptions, precedence constraints, etc. and γ is the objective function. If breakdowns are not present, the notation is $Jm \mid X_{ij} \sim H_{ij}(t)$; $\beta \mid \gamma$. This is a generalization of the single machine setting considered in Chapter 3.

5.3 Disjunctive Graph Model

The $J_m \mid\mid C_{max}$ problem can be represented with a disjunctive graph $G = (N, A, E)$ as shown by (Balas, 1969). With minor changes, this representation can also be used for the problems where completion time of each job should be calculated individually, rather than just the maximum (Pinedo, 2000). The set of nodes N contains one source

node U, one element for each operation (i, j), and n sink nodes. Source node U denotes the start of the schedule and sink node V_j represents the completion of job j. The set of conjunctive arcs $A = \{(i, j) \rightarrow (k, j)\}$ contains the arcs that connect the nodes representing each pair of consecutive operations (i, j) and (k, j) of job j. Each arc $(i, j) \rightarrow (k, j)$ has a length $| (i, j) \rightarrow (k, j) | = X_{ij}$ and represents the constraint that operation (k, j) may be started no less than X_{ij} time units after operation (i, j) has been started. Note that X_{ij} is a random variable. The node that represents the final operation of job j, say (h, j), has an arc of length X_{hj} incident to V_j. The source node U has n outgoing arcs, each one incident to the first operation of job j, $j = 1, \ldots, n$, with lengths equal to 0. Let N_i denote the set of nodes corresponding to the operations processed on machine i. The set of disjunctive arcs $E = \{(i, j) \leftrightarrow (i, k)\}$ has, for every pair of nodes (i, j) and (i, k) in N_i, two arcs going in opposite directions. The arc $(i, j) \rightarrow (i, k)$ has length $| (i, j) \rightarrow (i, k) | = X_{ij}$ and the arc $(i, k) \rightarrow (i, j)$ has length $| (i, k) \rightarrow (i, j) | = X_{ik}$. Each pair of disjunctive arcs represents the fact that two operations cannot be processed simultaneously on the same machine. Orienting a disjunctive arc pair in one direction or the other corresponds to a decision as to which operation comes before the other. For instance, fixing arc $(i, j) \rightarrow (i, k)$ implies that operation (i, k) is processed after operation (i, j). Figure 5.1 presents a 4-machine 3-job example.

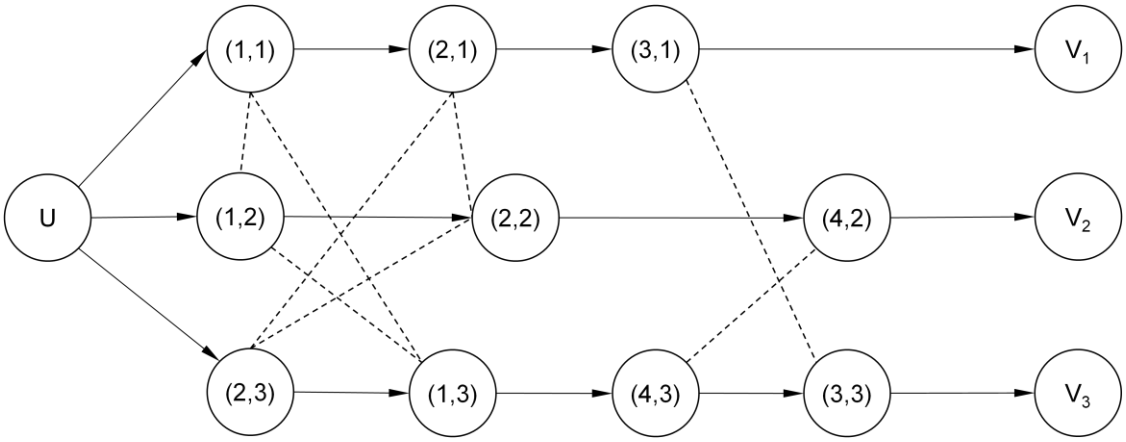


Figure 5.1. Disjunctive Graph Representation

Let $\sigma(E)$ denote a selection of disjunctive arcs from E. Any solution for the job shop problem is equivalent to some $\sigma(E)$, having exactly one arc from every disjunctive pair $(i, j) \leftrightarrow (i, k)$, such that the resulting graph $G(N, A, \sigma(E))$ is acyclic. Conversely, any selection $\sigma(E)$ satisfying the above properties corresponds to a feasible schedule. Let $L(O, O')$ denote the length of the critical (longest) path from node O to node O' in the graph $G(N, A, \sigma(B))$. If there is no path, then $L(O, O')$ is not defined.

The completion time C_j of job j is equal to $L(U, V_j)$. Recall that arc lengths are random variables; hence any path from U to V_j can be a critical path with some positive probability. Therefore, C_j is also a random variable.

5.4 Stability Measure (SM)

In the literature, schedule stability is generally measured in terms of the deviations in job completion times. Recall that job completion times are random variables. One can use their means as point estimators. In other words, it is reasonable to measure schedule stability in terms of deviations from *expected* completion times. A stable schedule in this sense is a schedule in which the difference between the realized completion times and the planned (i.e., mean) completion times are minimal. Specifically, the stability measure that is used in this chapter is

$$SM = E\left[\sum_{i=1}^{n}\left(E[C_i] - C_i\right)^2\right] = \sum_{i=1}^{n} Var[C_i]$$

Assume that a feasible job shop schedule and its disjunctive graph representation are given. To compute the objective function value of the feasible solution, one needs to calculate the variance of the completion time of each job. In the disjunctive graph representation, the completion time of job j is the length of the longest path from the source U to the sink V_j. The arc lengths (X_{ij}'s), however, are random variables. Hence, any path from U to V_j may be a longest path with some positive probability. Therefore, in order to analytically calculate the objective function

value, it is needed to enumerate all paths from U to V_j, to evaluate their probability of being a critical path, and finally to calculate the variance conditionally. Since there are an exponential number of paths and the calculation of the mentioned probabilities are challenging, the computation of the objective function cannot be achieved in polynomial time, which means that the problem is not in the class $\mathcal{NP}$. To alleviate this difficulty, the following surrogate measure is used instead of the real objective function.

Surrogate Stability Measure (SSM). *Variance of completion time of job j is estimated as the sum of variances of the arc lengths (b_{ij}'s) that lie on the critical (longest) path from U to V_j, where critical paths are calculated in terms of the expectations of the arc lengths (a_{ij}'s).*

Theorem 5.1. $J_m \mid X_{ij} \sim H_{ij}(t) \mid SSM$ *is* $\mathcal{NP}$*-hard* .

Proof. Consider the instance of $J_m \mid X_{ij} \sim H_{ij}(t) \mid SSM$ where variances of operation processing times are a constant multiple of their expectations. Any longest path in terms of expectations (called *longest expectation path* from now on) is also longest in terms of variances. Hence, the objective function can be computed as the sum of the variances of arcs that lie on longest variance paths. This makes the stochastic instance equivalent to a deterministic $J_m \mid\mid \sum C_j$ instance, where the processing times are taken as processing time variances of the $J_m \mid X_{ij} \sim H_{ij}(t) \mid SSM$ instance. $J_m \mid\mid \sum C_j$ is already known to be $\mathcal{NP}$-hard (Garey, Johnson and Sethi, 1976). This completes the proof. □

5.5 Branch-and-Bound (B&B) Algorithms

Note that it is possible to trivially minimize the objective function value by adding large blocks of idle times after the operations that lie on shortest variance paths (to make these paths coincide with the longest expectation paths). Such an action would,

however, deteriorate the performance of the schedule in terms of other common measures such as tardiness, makespan, flow time, etc. In this chapter, we confine ourselves to the schedules without unnecessary idle times.

We propose two branch-and-bound algorithms to solve the problem $J_m \mid X_{ij} \sim H_{ij}(t) \mid SSM$. Both algorithms use the same bounding scheme but differ on branching mechanism. The first branch-and-bound algorithm implicitly evaluates all schedules that are *active* whereas the second algorithm focuses on *non-delay* schedules.

Definition 5.1. *A schedule is* semi-active *if on any machine no operation can be processed earlier without changing the processing order of operations on that machine.*

Definition 5.2. *A schedule is* active *if on any machine no operation can be processed earlier without delaying another operation even with changing the processing order of operations on that machine.*

Definition 5.3. *A schedule is* non-delay *if no machine is kept idle when there is an operation is waiting for processing.*

Note that a non-delay schedule has to be active and similarly an active schedule has to be semi-active at the same time. The reverses, however, are not necessarily true.

An off-line schedule cannot be identified as active or non-delay without knowing the processing times in advance. Hence, in this chapter, the schedules are said to be active or non-delay with respect to mean processing times.

A node in both branch-and-bound trees consists of a partial schedule and its disjunctive graph representation. The graph in the root node includes only conjunctive

arcs (precedence constraints imposed by job routings). Operations are scheduled one at a time. Nodes that are deeper in the branching tree include more precedence constraints imposed by the disjunctive arcs whose orientations are decided. The partial schedule constituted by these precedence constraints develops into a complete feasible schedule at leaf nodes.

Before we present the branching schemes, more notation and terminology is needed. Operations are scheduled one at a time. An operation is *schedulable* if all the preceding operations within its job are already scheduled. Since there are *n.m* operations, the branch-and-bound trees have *n.m* levels. At level *t*, let

P_t be the partial schedule of scheduled operations;

S_t be the set of schedulable operations;

σ_k be the earliest mean time that the operation *k* in S_t could be started;

ϕ_k be the earliest mean time that the operation *k* in S_t could be finished.

The first branch-and-bound algorithm (active B&B) uses the following branching scheme due to Giffler and Thompson (1960). This scheme is modified slightly to generate non-delay schedules (Non-delay B&B). Specifically, a child node for operation *j* is created only if it is the earliest operation that is schedulable.

Active Branching Scheme:

$t \leftarrow 0$;
$P_0 \leftarrow \boldsymbol{null}$;
$S_0 \leftarrow$ (the set of all operations with no predecessors);
while ($t < n.m$) // there are operations to be scheduled
  $\phi^* \leftarrow \min_{k \in S_t}\{\phi_k\}$;
  $M^* \leftarrow$ (the machine on which ϕ^* occurs - in case of ties, choose arbitrarily);
  $O_t \leftarrow \{j \in S_t, |\sigma_j < \phi^*, j \; is \; processed \; on \; M^*\}$;
  foreach operation *j* **in** O_t
    Create a child node *n*; // in which *j* is the next scheduled operation within *n*

$P_{t+1} \leftarrow P_t + \{j\}$;
$S_{t+1} \leftarrow S_t \setminus \{j\} \cup \{k | k \; is \; immediate \; successor \; of \; j \; in \; its \; job\}$;
$t \leftarrow t + 1$;

Non-delay Branching Scheme:

$t \leftarrow 0$;
$P_0 \leftarrow \boldsymbol{null}$;
$S_0 \leftarrow$ (the set of all operations with no predecessors);
while ($t < n.m$) // there are operations to be scheduled
  $\sigma^* \leftarrow \min_{k \in S_t}\{\sigma_k\}$;
  $M^* \leftarrow$ (the machine on which σ^* occurs - in case of ties, choose arbitrarily);
  $O_t \leftarrow \{j \in S_t, |\sigma_j = \sigma^*, j \; is \; processed \; on \; M^*\}$;
  foreach operation *j* **in** O_t
    Create a child node *n*; // in which *j* is the next scheduled operation within *n*
      $P_{t+1} \leftarrow P_t + \{j\}$;
      $S_{t+1} \leftarrow S_t \setminus \{j\} \cup \{k | k \; is \; immediate \; successor \; of \; j \; in \; its \; job\}$;
      $t \leftarrow t + 1$;

Although the lower bound itself depends on the objective function and the implemented branching scheme, one property inherently holds in any branch-and-bound algorithm: the lower bound of a child node is greater than or equal to the lower bound of its parent node. Lower bounds that are used in the job shop scheduling literature generally based on the objective function value of the partial schedule or partial graph at a node. For $J_m \mid X_{ij} \sim H_{ij}(t) \mid SSM$, conventional lower bounds used in the literature do not have the aforementioned property. Inserting a new arc or operation into a parent node's partial graph to create its children may result in longer critical paths in the children, but these new critical paths may have lower total variance values. In other words, the objective function value of a partial schedule is not a lower bound for *SSM*, unlike for regular performance measures.

To calculate a lower bound of a node, we first examine the disjunctive arcs in the clique that belongs to the machine on which the inserted operation is processed. The arcs that identify the sequence of the scheduled operations on the machine are kept and the remaining (redundant) disjunctive arcs are permanently excluded from

the graph in the branching node, since these arcs cannot lie on a longest path. Figure 5.2 gives an example with 5 operations. In the figure, the clique corresponding machine i is examined. Since the order of operations on that machine is (1-2-3-4-5), the arcs that identify this order is kept (solid arcs), and the redundant ones (dashed arcs) are excluded. We then temporarily insert all the disjunctive arcs $(i, j) \leftrightarrow (i, k)$ that are not yet oriented, in both ways (i.e. both $(i, j) \rightarrow (i, k)$ and $(i, k) \rightarrow (i, j)$ are inserted). Using Dijkstra's algorithm, the shortest variance paths from U to V_j for all j, are identified in the augmented graph. The sum of the path variances from U to V_j for all j is a lower bound to *SSM*.

The performance of a branch-and-bound algorithm depends on the branching order. In our implementation, we use hybrid search strategy: a *best-first search* (the node with the best lower bound value is branched first) is used as long as the memory used to store unexplored nodes of the branch-and-bound tree is below a threshold value, T_1.

The search strategy is switched to a *depth-first search* (the most recent node is branched first) until the memory requirement becomes less than another threshold value, T_2, where $T_2 < T_1$. After this point in time, best-first search is back in use and

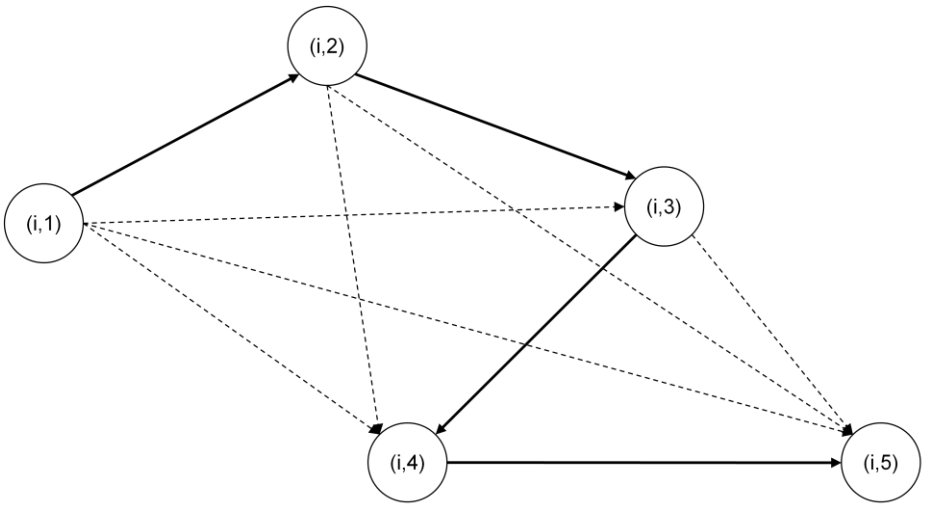
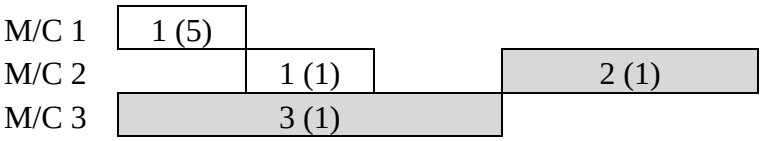


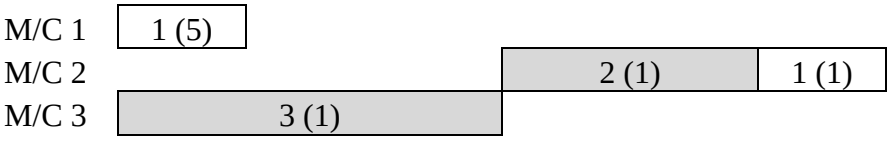
Figure 5.2. Examining of Disjunctive Arcs on a Machine Clique

exploration of nodes continues in that fashion until an optimal solution is found or the time limit is up.

Note that if the objective function is a regular performance measure, it can be easily proven that an active schedule that is optimal exists. As mentioned earlier, *SSM* can be improved by making jobs complete later. In other words, *SSM* is not a regular performance measure and a better feasible solution may exist in the class of semi-active schedules. The following small numerical example illustrates this. Consider a job shop with two jobs and three machines. Job 1 first receives its processing on machine 1 and then on machine 2 while job 2 has to be processed first on machine 3 and then on machine 2. Two feasible schedules are given in Figure 5.3. In the figure, operations of job 2 are depicted in gray. The numbers in the center of rectangles that denote operations are means and variances of the processing times, respectively. Recall that *SSM* value of a schedule is the sum of variance of the processing times of the operations that lie on the longest path from source to the sinks in the disjunctive graph representation.



a. An Active Schedule



b. A Semi-Active Schedule

Figure 5.3. Numerical Example

It is not difficult to see that the *SSM* value of the active schedule given in Figure 5.3.a is (5 + 1) + (1 + 1) = 8. The objective function value of the semi-active schedule given in Figure 5.3.b is (1 + 1 + 1) + (1 + 1) = 5. As it can also clearly be seen from this example, contrary to the regular performance measures, an optimal *SSM* value is not necessarily attained by an active schedule.

Unfortunately, our pilot computational tests indicate that a branch-and-bound algorithm that implicitly enumerates all semi-active schedules requires too much computation time to be of practical value. Therefore, the search space is restricted to the classes of active and non-delay schedules.

5.6 A Beam-Search (BS) Algorithm

The proposed branch-and-bound algorithm becomes increasingly expensive in terms of computational time as the problem size gets larger. In this section, we develop a beam-search algorithm that can be used to solve large problems.

As mentioned in Section 3.7, beam search is an approximate branch-and-bound method which operates on a search tree. Similar to the previous case, we again use dependent beams (i.e., at each level, all the descendants are evaluated and the best β of them are chosen without paying attention to their ancestors). Operations are scheduled one at time in a constructive manner, like in a branch-and-bound algorithm. Specifically, we first generate all the children of all the nodes in the current level using the active branching scheme explained in Section 5.5. Each child is then temporarily completed using four different dispatching rules to globally evaluate its performance.

The dispatching rules are SVPT (Smallest Variance of Processing Time first), SCV (Smallest Coefficient of Variation first), SEPT (Shortest Expected Processing Time first), and LEPT (Longest Expected Processing Time first). The objective function value (*SSM*) under each completion scheme is calculated, and the global evaluation function value of the child is taken as the maximum of them. The best (i.e., with smallest global evaluation function values) β children are kept and the next

iteration begins. The rationale behind this minimax type of global evaluation is to be able to identify the partial schedules of first few operations that yield good *SSM* values, no matter how the rest of the schedule is completed. A minimax global evaluation function is conjectured to help avoid inferior starting partial schedules that one can get stuck into because of the inherent myopic nature of the heuristic. Our pilot computational experiments also indicate that the minimax type explained above outperforms a minimin global evaluation function.

5.7 A Tabu-Search (TS) Algorithm

To further increase the quality of the solutions that are obtained by the beam-search heuristic, we propose a tabu-search (TS) algorithm.

We start with five seed schedules and generate their neighborhood. At each iteration of TS, we evaluate the objective function value of the generated schedules and adopt the best schedule (if not tabu) as a new seed to the neighborhood generator. The generator creates new schedules, which in turn are evaluated again. The search continues in this fashion until the stopping criterion is met. The neighborhood generator reverses the orientation of a disjunctive arc on a longest expectation path to obtain a neighbor. Note that the neighbors generated with this move cannot be infeasible (otherwise the reversed arc would not be part of a longest expectation path). The reversed arc is added to the tabu list to prevent immediate backtracking. If the best neighbor performs better than the current best solution so far, it is taken as a new seed, even if the move needed to generate it is tabu (*aspiration criterion*).

The initial seeds are generated using SVPT (Smallest Variance of Processing Time first), SCV (Smallest Coefficient of Variation first), SEPT (Shortest Expected Processing Time first), LEPT (Longest Expected Processing Time first) dispatching rules and the beam-search algorithm explained in the previous section.

5.8 Computational Experiments

To assess the quality of the proposed algorithms (active B&B, non-delay B&B, beam-search, and tabu-search), several input problems are solved using each. Since the objective function under study is the total variance on the longest expectation path, long arcs with small variances are likely to be included in critical paths of an optimal solution. It is expected to take longer time to solve the problem optimally if fewer such arcs exist. In other words, the difficulty of the problem instances is conjectured to be dependent on the ratio of expectations and variances of the arcs. To investigate the validity of this conjecture, a computational test bed is prepared to include three levels of coefficient of variations for the processing times of the operations. The processing time means (μ) are sampled from a discrete uniform distribution with parameters 40 and 60. The coefficient of variations are sampled from U[0, 0.4], U[0.8, 1.2] and U[1.8, 2.2], and called levels *CV1*, *CV2* and *CV3*, respectively.

The effects of machine routings are also examined. All jobs have operations on all machines. Three levels of routing are considered. On one extreme, all jobs are taken to visit the machines in the same (flow shop) order (*fixed* routing). On the other extreme all routings are randomly generated (*random* routing). Between these two extremes, a *semi-random* routing is also considered, where half of the operations of every job are processed in the same machine order.

In this chapter, a problem instance is called to be of size n x m, if the number of jobs is n and the number of machines is m. In our experiments, four levels of problem size are considered: 5 x 5, 5 x 10, 10 x 5, and 10 x 10.

To sum up, the computational environment consists of 36 problem classes (4 levels of size x 3 levels of coefficient of variation x 3 levels of machine routing). 10 instances of each problem class are generated, resulting in a test bed of 360 instances.

All algorithms are coded in the C++ language and run on a Linux box running Debian Etch on a Pentium 4 2.4GHz CPU with 512MBs of physical memory. The threshold values T_1 and T_2 for branch-and-bound algorithms are set to 80% and 90% of available physical memory, respectively.

5.8.1 Cases with No Breakdown

5.8.1.1 Branch-and-Bound Algorithms

A preliminary experimentation indicated that a branching scheme that generates all semi-active schedules is computationally impractical even for 5 x 5 problems. In our computational experiments we consider two branch-and-bound algorithms (active B&B and non-delay B&B), both of which use operation insertion scheme. Note that the search space of active B&B is a superset of that of non-delay B&B. The reason why non-delay B&B is included is to be able to examine the trade-off between computational time spent and solution quality obtained when search space is reduced in size. A maximum of two hours of computational time (7200 CPU seconds) is allowed for each instance. The results are given in Tables 5.2 and 5.3.

In Tables 5.2 and 5.3, levels of machine routing are shown in columns and the rows list the levels of coefficient of variation. The four numbers in each cell report the lower bound, the upper bound, the percentage relative gap and the solution time for the corresponding problem class. The numbers are the averages of the *SSM* values for 10 instances in that class.

On examining Tables 5.2 and 5.3, we observe that the impact of an increase in the number of jobs is more than the impact of an increase in the number of machines on the solution time.

For 5-job problems, less computational time is needed to solve the instances with random machine routings than the instances with fixed or semi-random routings. Similar results are also reported in the literature for regular performance measures (Singer and Pinedo, 1998).

For 10-job problems, two hours of CPU time is not enough to arrive at optimality but it can be said that the solution quality for the flow shop problems is better in terms of average relative gap.

Table 5.2. Summary of Results for Active B&B

		ACTIVE B&B					
		5x5			5x10		
		fixed	semi	random	fixed	semi	random
CV1	Lower Bound	2065.8	2184.5	1916.4	4206	1443	**4593.6**
	Objective	2065.8	2184.5	1916.4	4591.8	4842.8	4593.6
	Gap (%)	0.00%	0.00%	0.00%	9.17%	235.61%	0.00%
	Time (s)	10.48	40.209	7.094	5565.676	7200.035	80.846
CV2	Lower Bound	84674	85335.6	79497.1	127484.5	35888.7	123518.6
	Objective	84674	85335.6	79497.1	135586.1	135232.6	123518.6
	Gap (%)	0.00%	0.00%	0.00%	6.35%	276.81%	0.00%
	Time (s)	23.923	47.57	8.689	7200.01	7200	86.523
CV3	Lower Bound	343356.6	352073.7	322804.2	519907.5	146625	514031.8
	Objective	343356.6	352073.7	322804.2	560549.1	547626.8	514031.8
	Gap (%)	0.00%	0.00%	0.00%	7.82%	273.49%	0.00%
	Time (s)	26.034	47.26	9.047	7200	7200	89.464

		10x5			10x10		
		fixed	semi	random	fixed	semi	random
CV1	Lower Bound	4918.9	2244.8	1929.5	8005.3	2485.1	2386.7
	Objective	14107.4	13805	11360.1	21208.5	20025.5	18295.7
	Gap (%)	186.80%	514.98%	488.76%	164.93%	705.82%	666.57%
	Time (s)	7200	7200	7200	7200	7200	7200
CV2	Lower Bound	127298	60332.8	55358.7	246798	64594.6	74798.3
	Objective	315417.9	292443.5	237489.8	423422.4	420699	429426.9
	Gap (%)	147.78%	384.72%	329.00%	71.57%	551.29%	474.11%
	Time (s)	7200	7200	7200	7200	7200	7200
CV3	Lower Bound	527654.8	257412.5	221284.4	968543.3	260279	317755.1
	Objective	1261260	1183611	986705.5	1696313	1665424	1714593
	Gap (%)	139.03%	359.81%	345.90%	75.14%	539.86%	439.60%
	Time (s)	7200	7200	7200	7200	7200	7200

Table 5.3. Summary of Results for Non-Delay B&B

		NON-DELAY B&B					
		5x5			5x10		
		fixed	semi	random	fixed	semi	random
CV1	Lower Bound	2827.7	3132.5	3006.5	5683.4	5673.2	6207.2
	Objective	2827.7	3132.5	3006.5	5683.4	5673.2	6207.2
	Gap (%)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Time (s)	0.216	0.219	0.08	1.161	0.251	0.028
CV2	Lower Bound	84713	87370.3	82053.8	136793.9	133433.1	130697.3
	Objective	84713	87370.3	82053.8	136793.9	133433.1	130697.3
	Gap (%)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Time (s)	0.229	0.214	0.066	1.366	0.245	0.028
CV3	Lower Bound	343527.4	355611	323515.5	561562.3	548518.4	522670.5
	Objective	343527.4	355611	323515.5	561562.3	548518.4	522670.5
	Gap (%)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	Time (s)	0.258	0.217	0.076	1.417	0.234	0.028

		10x5			10x10		
		fixed	semi	random	fixed	semi	random
CV1	Lower Bound	4284.2	2197.3	2531.3	5459.4	2326	3025.5
	Objective	6745.5	10568.4	9174.6	15899.2	16869.5	14676
	Gap (%)	57.45%	380.97%	262.45%	191.23%	625.26%	385.08%
	Time (s)	7200	7200	7200	7200	7200	7200
CV2	Lower Bound	120199.8	56474.4	75299.8	230397.6	75028	95304.6
	Objective	213633.3	268408	203149.3	383282.1	390996	374896.9
	Gap (%)	77.73%	375.27%	169.79%	66.36%	421.13%	293.37%
	Time (s)	7200	7200	7200	7200	7200	7200
CV3	Lower Bound	501824.2	229035.9	306655.5	943383.9	293856	383167.8
	Objective	946032.2	1062296	854715.1	1563216	1545744	1506333
	Gap (%)	88.52%	363.81%	178.72%	65.70%	426.02%	293.13%
	Time (s)	7200	7200	7200	7200	7200	7200

Observe that the lower bound values for flow shop problems are significantly higher than those for the problems with random machine routing for 10-job problems.

We note that for small problems, as coefficient of variation increases the computational time needed to solve the problems also increase. The relative gap generally decreases with increasing coefficient of variation levels. This effect is more visible for active B&B. This observation could be explained by the intuition that as coefficient of variation increases, total variances on shortest variance paths get larger, the lower bounds increase, and the relative gaps decrease as a result.

It can be seen that non-delay B&B runs remarkably faster than active B&B as expected. The performance of active B&B is better for small problems in terms of the optimal objective function value. As for large problems, where two hours of CPU time is not enough to reach optimality, the performance of non-delay B&B improves and it even outperforms active B&B for the majority of the problem classes, both in terms of relative gap and the upper bound value. This is intuitive because non-delay B&B algorithm is able to search a larger portion of its search space, compared to the active B&B.

5.8.1.2 Heuristics

Beam-Search Algorithm:

The proposed beam-search algorithm is compared with four dispatching rules to assess its quality. The dispatching rules are SVPT, LEPT, SCV, and SEPT. After our pilot experiments, we decide to use a beam width of size 25 for all instances. The results are summarized in Table 5.4. The numbers in the cells in Table 5.4 are the averages of the *SSM* of the instances in the corresponding problem class.

On examining Table 5.4, we observe that the proposed beam-search algorithm is significantly better than all dispatching rules for all problem classes. In general, SVPT and SEPT dispatching rules are competitive and are better than LEPT and SCV rules.

Table 5.4. Summary of Results for Beam-Search and Dispatching Rules

		5x5			5x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	4440,8	4395,4	3601,9	8335,5	8510,4	7125,8
	LEPT	5798,0	5166,9	5016,1	9679,4	9477,2	8827,9
	SCV	6017,0	6820,3	5268,8	10180,6	10001,0	8932,9
	SEPT	4489,0	4593,9	4475,6	8330,3	9266,0	6594,9
	BS	**3139,8**	**2975,5**	**2539,5**	**5836,5**	**6010,5**	**5147,7**
CV2	**SVPT**	103920,6	105589,8	94439,2	154800,7	158671,1	140297,1
	LEPT	108198,7	107682,5	105236,3	172221,7	174342,7	150482,9
	SCV	118375,6	112229,9	103566,6	179323,9	168771,3	151181,2
	SEPT	96730,0	103316,1	97368,2	152693,4	174122,9	128901,8
	BS	**88121,9**	**89708,0**	**85893,9**	**142634,8**	**139268,8**	**125568,0**
CV3	**SVPT**	408332,0	421912,3	390256,7	637576,9	632323,4	578252,3
	LEPT	442377,5	435167,3	407903,6	694006,2	699064,5	585851,1
	SCV	455519,2	455505,2	386366,3	690865,6	678466,0	592451,7
	SEPT	384797,1	409327,9	388134,5	607582,4	694116,4	522566,1
	BS	**352591,9**	**369848,7**	**345498,6**	**579267,0**	**569585,4**	**519805,0**

		10x5			10x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	15018,0	16185,1	14046,5	23553,3	25638,0	19636,8
	LEPT	17588,5	17899,4	17455,8	27100,6	24017,4	29017,7
	SCV	21777,7	21216,4	18586,3	29428,4	27721,2	26415,3
	SEPT	18658,3	17274,0	15520,2	25227,8	24710,2	22231,9
	BS	**10849,2**	**11568,5**	**8754,6**	**17503,9**	**15884,9**	**14657,7**
CV2	**SVPT**	316362,0	294050,8	256432,7	502808,1	455758,6	440502,4
	LEPT	305645,4	308165,5	295302,5	504190,1	474628,2	466878,8
	SCV	346579,8	334703,2	276054,1	509182,3	497287,5	478893,7
	SEPT	334363,2	307731,2	270848,8	464088,4	453322,7	421477,6
	BS	**252815,9**	**258006,8**	**223324,4**	**407599,1**	**400426,5**	**374031,0**
CV3	**SVPT**	1279427,0	1232813,0	1099976,0	1888674,0	1834710,0	1800465,0
	LEPT	1230091,0	1237944,0	1177658,0	1964916,0	1881757,0	1857110,0
	SCV	1330593,0	1296854,0	1146025,0	1971870,0	1949900,0	1919373,0
	SEPT	1334445,0	1244550,0	1100796,0	1821795,0	1793120,0	1681552,0
	BS	**1030773,4**	**1065955,0**	**950926,3**	**1616278,0**	**1615757,0**	**1540526,0**

Tabu-Search Algorithm:

Recall that the initial seeds for the TS algorithm are generated using the SVPT, SCV, SEPT and LEPT dispatching rules and the beam-search algorithm. After some pilot experimentation, it is decided to allow a maximum of 10000 (40000) iterations and to use a tabu list of length 10 (40) for problems with 5 (10) jobs. The results are summarized in Table 5.5.

Levels of machine routing are shown in columns and the rows list the levels of coefficient of variation and five initial solutions (SVPT, LEPT, SCV, SEPT and BS). The numbers in each cell report the average *SSM* value of the corresponding problem class and initial solution. The values in the parentheses are the number of times in which the corresponding initial solution yields the best *SSM* value. We observe that the average objective function values are close to each other and all initial solutions are competitive. In increasing the quality of the tabu-search algorithm, each initial solution method has its own contribution and none of them is dominant to another.

Note that tabu-search algorithm is not restricted to the classes of active schedules. Although all five initial schedules are active, the act of reversing disjunctive arcs on critical paths (i.e., the neighborhood function) does not necessarily generate another active schedule. Thus, tabu-search algorithm can potentially result in schedules with objective function values better than *all* active schedules, which is the set that contains the optimal solutions of the proposed branch-and-bound algorithms. To assess the quality of the tabu-search algorithm, we compare it to the proposed branch-and-bound algorithms. The summary results are given in Table 5.6. In Table 5.6, levels of machine routing are shown in columns and the rows list the levels of coefficient of variation, the three alternative algorithms (active B&B, non-delay B&B, and TS) and the best known upper bound. The numbers in cells are the averages of the *SSM* values of the corresponding problem class. The percentages are the average relative gaps of the corresponding algorithm's upper bound with respect to the best known objective function. The values in the parentheses are the number of times in which the corresponding algorithm yields the best known *SSM* value.

Table 5.5. Summary of Results for Tabu-Search

		5x5			5x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	**1915 (5)**	**2152,4 (5)**	1895,1 (8)	4999,8 (2)	4353,7 (3)	4574,4 (5)
	LEPT	1959,7 (5)	2182 (6)	1954,8 (5)	5161,7 (1)	4373,8 (4)	4537,5 (5)
	SCV	2144 (1)	2186,4 (5)	1943,1 (6)	4949,3 (3)	4342,1 (4)	4659,3 (3)
	SEPT	2286,8 (2)	2192,4 (6)	**1886,2 (9)**	**4696,8 (5)**	4387,2 (5)	4588,6 (4)
	BS	2020,8 (5)	2189,8 (8)	1953,1 (5)	4937 (3)	**4333,5 (3)**	**4522,6 (3)**
CV2	**SVPT**	**85646 (9)**	**85750,1 (9)**	79753,2 (8)	139726,6 (2)	135464,1 (2)	124140 (9)
	LEPT	89761,8 (5)	86263,5 (7)	79750,1 (9)	146513,2 (0)	136075,2 (4)	124573,3 (8)
	SCV	87956,7 (4)	87439,8 (6)	79978,8 (6)	142704,4 (3)	133673 (3)	124263,4 (8)
	SEPT	86478,4 (7)	**85750,1 (9)**	79936,1 (7)	**138179,1 (5)**	**131716,4 (5)**	**123796,8 (9)**
	BS	86587,4 (6)	85942,9 (8)	**79473,5 (8)**	139843 (1)	132365,7 (4)	123895,1 (9)
CV3	**SVPT**	349859,4 (4)	354836,2 (7)	**322251,8 (10)**	572940,4 (3)	545018 (5)	516041,7 (7)
	LEPT	353616,6 (2)	355024,5 (8)	**322251,8 (10)**	579668,1 (1)	559331,9 (4)	516169 (7)
	SCV	358474,6 (1)	354790,9 (5)	**322251,8 (10)**	579731,1 (2)	554955,9 (3)	**514233 (9)**
	SEPT	351438,6 (3)	355163,7 (7)	**322251,8 (10)**	**570057,8 (5)**	545233,2 (7)	515554,9 (6)
	BS	**347262,4 (4)**	**352974,7 (8)**	**322251,8 (10)**	571673,2 (4)	**544987,6 (5)**	515991,4 (7)

		10x5			10x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	8471,7 (0)	6994 (0)	**5693,5 (3)**	13282,1 (1)	10846,2 (0)	**9838,2 (3)**
	LEPT	8175,1 (1)	6663,8 (5)	5777,5 (1)	13549,6 (0)	10684,7 (2)	10112,3 (2)
	SCV	8080,4 (1)	6825,8 (1)	5785,1 (1)	13461,3 (2)	10669,4 (3)	10161,9 (2)
	SEPT	7808,4 (1)	**6639 (2)**	5695,4 (3)	13434,3 (2)	**10529,6 (3)**	10119,4 (2)
	BS	**7562 (7)**	6740,3 (2)	5710,2 (2)	**12491,7 (5)**	10912,2 (2)	9968,4 (1)
CV2	**SVPT**	236651,5 (1)	218717 (3)	196915,5 (2)	391789 (2)	**358601 (2)**	**329806,6 (5)**
	LEPT	235783,7 (3)	221306 (0)	198279,4 (3)	393408,2 (1)	360005,8 (5)	336647,7 (1)
	SCV	235938 (2)	**218526 (4)**	197267,1 (2)	393218,4 (1)	362060 (2)	335277,5 (0)
	SEPT	**235033,4 (2)**	221985,1 (0)	**195499,5 (2)**	390106,1 (4)	363667,1 (1)	334723,3 (2)
	BS	237069 (2)	218597,8 (3)	198397,1 (1)	**387384,3 (2)**	360030,2 (0)	333950,6 (2)
CV3	**SVPT**	986662,3 (1)	931396,5 (3)	855423,1 (3)	1586115 (2)	1486689 (4)	1359234 (2)
	LEPT	**983795,3 (2)**	**928087,9 (3)**	856621,7 (1)	1596043 (0)	1481072 (3)	1363112 (2)
	SCV	988433,8 (1)	931782 (1)	855823,6 (2)	1591653 (1)	1493451 (2)	1372479 (0)
	SEPT	988388,3 (3)	931472,7 (1)	**849961,5 (3)**	1579024 (3)	1495516 (0)	**1358426 (6)**
	BS	983944,4 (3)	931350,9 (2)	857460,4 (1)	**1568253 (4)**	**1479521 (1)**	1382277 (0)

Table 5.6. Comparison of Tabu Search and Branch-and-Bound Algorithms

		5x5								5x10							
		fixed			semi			random		fixed			semi			random	
CV1	Active	2066	15,81%	(1)	2185	3,83%	(6)	1916	1,75%	(7)	4592	6,16%	(4)	4843	17,76%	(0)	4594	5,06%	(2)
	Non-delay	2828	58,52%	(0)	3133	48,88%	(0)	3007	59,63%	(1)	5683	31,40%	(1)	5673	37,95%	(1)	6207	41,97%	(0)
	TS	1784	0,00%	(10)	2104	0,00%	(10)	1883	0,00%	(10)	4426	2,33%	(7)	4113	0,00%	(10)	4387	0,35%	(9)
	Best	1784			2104			1883			4325			4113			4372		
CV2	Active	84674	0,00%	(10)	85336	0,00%	(10)	79497	0,22%	(9)	135586	0,00%	(10)	135233	4,16%	(1)	123519	0,00%	(10)
	Non-delay	84713	0,05%	(9)	87370	2,38%	(3)	82054	3,44%	(2)	136794	0,89%	(7)	133433	2,77%	(5)	130697	5,81%	(0)
	TS	85604	1,10%	(7)	85505	0,20%	(8)	79324	0,00%	(10)	136568	0,72%	(6)	130608	0,59%	(7)	123773	0,21%	(7)
	Best	84674			85336			79324			135586			129837			123519		
CV3	Active	343357	0,00%	(10)	352074	0,00%	(10)	322804	0,17%	(9)	560549	0,00%	(10)	547627	1,14%	(2)	514032	0,00%	(10)
	Non-delay	343527	0,05%	(9)	355611	1,00%	(5)	323516	0,39%	(7)	561562	0,18%	(8)	548518	1,30%	(2)	522671	1,68%	(2)
	TS	344819	0,43%	(7)	352203	0,04%	(9)	322252	0,00%	(10)	565004	0,79%	(5)	541856	0,07%	(8)	514032	0,00%	(10)
	Best	343357			352074			322252			560549			541464			514032		

Table 5.6. Comparison of Tabu Search and Branch-and-Bound Algorithms Cont'd

		10x5						10x10					
		fixed		semi		random		fixed		semi		random	
CV1	Active	14107	116,21% (0)	13805	118,53% (0)	11360	107,92% (0)	21209	75,19% (0)	20026	104,02% (0)	18296	94,04% (0)
	Non-delay	6746	3,38% (7)	10568	67,30% (0)	9175	67,92% (0)	15899	31,34% (0)	16870	71,87% (0)	14676	55,65% (0)
	TS	7159	9,72% (3)	6317	0,00% (10)	5464	0,00% (10)	12106	0,00% (10)	9815	0,00% (10)	9429	0,00% (10)
	Best	6525		6317		5464		12106		9815		9429	
CV2	Active	315418	47,64% (0)	292444	36,56% (0)	237490	22,96% (0)	423422	12,70% (0)	420699	19,81% (0)	429427	31,60% (0)
	Non-delay	213633	0,00% (10)	268408	25,33% (0)	203149	5,18% (0)	383282	2,02% (3)	390996	11,35% (0)	374897	14,89% (0)
	TS	231703	8,46% (0)	214154	0,00% (10)	193148	0,00% (10)	380023	1,15% (7)	351126	0,00% (10)	326308	0,00% (10)
	Best	213633		214154		193148		375708		351126		326308	
CV3	Active	1261260	33,44% (0)	1183611	29,13% (0)	986706	17,54% (0)	1696313	10,07% (0)	1665424	14,43% (0)	1714593	27,65% (0)
	Non-delay	946032	0,09% (9)	1062296	15,90% (0)	854715	1,82% (0)	1563216	1,43% (3)	1545744	6,20% (0)	1506333	12,15% (0)
	TS	967171	2,33% (1)	916594	0,00% (10)	839477	0,00% (10)	1556121	0,97% (7)	1455469	0,00% (10)	1343188	0,00% (10)
	Best	945171		916594		839477		1541133		1455469		1343188	

On examining Table 5.6, we observe that the proposed tabu-search algorithm performs quite well and in general yields better solutions than active optimal schedules. Especially for large problems, it is impractical to solve problems to optimality and the proposed tabu-search algorithm generates the most promising schedules. We conclude that the proposed tabu-search algorithm can be used to generate stable schedules.

5.8.2 Breakdown and Repair Cases

In this section we include a breakdown / repair process. All machines are subject to random breakdowns. We assess the performance of the proposed heuristics (beam-search and tabu-search) both under mild and heavy breakdowns. We use Gamma distribution as a busy-time distribution with a shape parameter of 0.7, and a scale parameter that is arranged so that the mean busy-time is 400 for mild breakdowns and 200 for heavy breakdowns, respectively. We use Gamma distribution with a shape parameter of 1.4 for the down-time distribution, as recommended by Law and Kelton (2000). The scale parameter of the down-time distribution is arranged to have a mean repair duration of 50.

Recall that the surrogate measure *SSM* estimates stability as the sum of arc variances on the longest expectation paths. However, in the presence of a breakdown/repair process, it is difficult to calculate *SSM* analytically because one does not know which operations will be interrupted in advance. We use the common approach of inflating the processing times of the operations appropriately to account for the effects of breakdowns (e.g., Mehta and Uzsoy, 1998). Specifically, we pre-process the problem instance and modify the means and the variances of operation durations as follows:

$$\mu_{ij} = a_{ij} \times \left(1 + \frac{E[D_i]}{E[U_i]}\right)$$

$$\sigma_{ij}^2 = b_{ij} + \left(\frac{a_{ij}}{E[U_i]} \times V[D_i]\right)$$

where D_i and U_i are the independent and identically distributed random variables denoting down and up times for machine i, respectively and *E[.]* and *V[.]* are the expectation and the variance operators. The mean and the variance of the processing time of operation ij is taken as μ_{ij} and σ_{ij}^2. All the algorithms (beam-search, tabu-search and dispatching rules) work as if no breakdowns occur, except that input mean and variance values are inflated. Since the *SSM* values used by these algorithms now become estimates, we compare the performance of the mentioned algorithms by simulating the generated schedules to approximate total completion time variances (*SM* itself). Generated schedules are simulated 100 times. During the simulations, first the processing times of operation ij is sampled from a Gamma distribution with mean a_{ij} and variance b_{ij}. Then breakdown times and repair durations are inserted into the schedule. Finally, job completion times are recorded and their variances are calculated. Sum of completion time variances are taken as the performance measure (*SM*).

5.8.2.1 Results of Heuristics

Beam-Search Algorithm:

The proposed beam-search algorithm is compared with four dispatching rules to assess its quality under random machine breakdowns. The dispatching rules are SVPT, LEPT, SCV, and SEPT as in the no-breakdown case.

The results are summarized in Tables 5.7 and 5.8. The numbers in the cells in these tables are the averages of the simulated *SM* values of the instances in the corresponding problem class.

On examining Tables 5.7 and 5.8, we observe that the proposed beam-search algorithm is better than all dispatching rules. In general, SVPT and SEPT dispatching rules are competitive and are better than LEPT and SCV rules even though the differences are not as significant as in the no-breakdown case.

Table 5.7. Simulation of Results for Beam-Search and Dispatching Rules under Mild Breakdowns

		5x5			5x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	23933.9	17029.2	17102.0	29361.3	18060.5	16188.5
	LEPT	24028.5	17707.8	17863.0	28443.9	21598.4	19759.9
	SCV	23494.0	19655.5	17239.0	32967.7	21624.4	18096.5
	SEPT	22468.2	**15127.7**	18177.3	25613.4	20791.3	16506.4
	BS	**18839.6**	15542.3	**15801.2**	**24773.1**	**17293.7**	**15861.3**
CV2	**SVPT**	143466.3	135662.9	126200.5	200843.9	193694.4	163981.0
	LEPT	151246.0	146733.9	139251.1	216107.3	203890.7	169288.4
	SCV	167098.5	143091.2	137980.2	225649.7	199210.4	170734.9
	SEPT	129745.2	128522.8	132998.0	**190624.0**	200066.2	**153153.4**
	BS	**123109.8**	**124030.4**	**119934.0**	190713.4	**170786.7**	157559.9
CV3	**SVPT**	669553.7	669898.8	658316.9	971082.6	1024172.0	865921.6
	LEPT	759310.6	695262.6	703386.9	1074895.1	1123339.4	932571.4
	SCV	750824.7	766183.9	647982.6	1077075.2	1079013.7	931304.8
	SEPT	635106.0	643761.1	667179.6	909692.6	1089610.9	**831062.7**
	BS	**593311.5**	**617157.0**	**598999.1**	**902078.2**	**915595.6**	843118.9

		10x5			10x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	94293.9	85114.0	63250.0	109065.0	84996.2	67537.6
	LEPT	84507.5	**74087.2**	74214.4	106911.1	79781.8	80681.4
	SCV	108525.8	85199.5	63560.0	116191.3	86241.1	73006.8
	SEPT	104113.6	74665.5	**60487.5**	106884.6	82828.4	70879.3
	BS	**73429.4**	82522.6	63833.9	**86469.5**	**66513.9**	**68738.4**
CV2	**SVPT**	398941.7	376591.7	339533.6	596479.8	523274.6	548588.9
	LEPT	391673.0	385181.4	381700.6	611405.1	498636.5	574515.5
	SCV	458122.9	417916.2	354950.9	598590.2	568547.4	600349.0
	SEPT	437702.8	376545.7	348099.8	534023.9	528291.7	533655.2
	BS	**344188.2**	**332350.4**	**321656.1**	**495112.8**	**479477.0**	**470180.1**
CV3	**SVPT**	2159780.0	1978826.0	1932302.0	2831443.0	2750347.0	2645370.0
	LEPT	2159196.0	2099690.0	1975625.0	3021299.0	2670930.0	2818513.0
	SCV	2358755.0	2268781.0	2030677.0	2800855.0	2890743.0	2905333.0
	SEPT	2290540.0	2045607.0	1912175.0	2691100.0	2683293.0	2619190.0
	BS	**1998864.0**	**1921427.0**	**1818861.0**	**2524301.0**	**2511141.0**	**2524559.0**

Table 5.8. Simulation of Results for Beam-Search and Dispatching Rules under Heavy Breakdowns

		5x5			5x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	49590.5	40756.1	43224.2	56835.8	46156.9	**36480.1**
	LEPT	49431.1	46282.0	42373.0	53361.5	47773.7	41961.3
	SCV	48051.4	49327.3	41958.3	68181.6	47993.0	39952.9
	SEPT	44475.2	36627.5	43065.5	**47942.6**	43885.0	36964.1
	BS	**40958.6**	**35678.9**	**40448.9**	55160.4	**35444.7**	41133.1
CV2	**SVPT**	194598.6	185528.3	167359.1	240919.4	257803.5	219398.8
	LEPT	213622.7	208174.2	188262.0	236156.1	270014.2	238981.3
	SCV	221359.8	207598.5	181114.6	287899.5	262595.3	236364.2
	SEPT	179142.7	177982.9	172994.6	241135.8	281012.4	217251.3
	BS	**175508.9**	**176617.3**	**164756.3**	**226526.9**	**222416.3**	**213198.2**
CV3	**SVPT**	845594.6	877882.2	788388.7	1229015.8	1169418.1	1214716.2
	LEPT	871107.7	998969.6	800856.6	1328387.0	1243003.1	1268941.0
	SCV	917430.4	1002427.0	817510.7	1354237.2	1256490.4	1224225.7
	SEPT	813134.8	**856398.7**	830315.5	**1173682.8**	1302198.3	1187152.2
	BS	**776816.3**	886781.1	**719401.6**	1174791.6	**1077531.7**	**1169419.0**

		10x5			10x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	231464.9	191170.7	170512.4	271900.2	261005.2	238510.6
	LEPT	189207.2	**177440.9**	192610.6	281369.9	**228686.0**	236687.3
	SCV	252928.2	195163.5	**164529.8**	308634.5	269339.1	247341.9
	SEPT	248603.1	182564.5	171980.6	274506.4	259419.7	**219687.6**
	BS	**160780.7**	194858.6	169572.3	**223305.5**	238576.5	226393.6
CV2	**SVPT**	609751.6	523893.6	488495.8	853268.2	810490.3	787013.1
	LEPT	597109.3	552993.7	568058.1	895818.8	818009.6	813152.7
	SCV	719258.8	564390.0	565223.2	868114.2	881592.0	869948.9
	SEPT	668045.9	513344.7	486194.1	774942.9	886272.1	783096.9
	BS	**512605.0**	**475232.1**	**461917.4**	**715352.2**	**743952.2**	**765704.2**
CV3	**SVPT**	2613689.0	2563227.0	2645420.0	3820419.0	3796737.0	4078698.0
	LEPT	2826474.0	2665467.0	2728483.0	3858700.0	3886304.0	4137995.0
	SCV	2904115.0	2840645.0	2583241.0	3968915.0	4192741.0	4186344.0
	SEPT	2775035.0	2537331.0	2536759.0	3760281.0	3799653.0	3916455.0
	BS	**2414115.0**	**2387027.0**	**2370191.0**	**3463775.0**	**3657930.0**	**3560951.0**

For all four dispatching rules and beam-search algorithm, the objective function used by the algorithms (*SSM* value with the inflated processing times) and the simulation results are compared in a correlation study. For all algorithms, the correlation coefficients between *SSM* values and *SM* values are found to be larger than 0.97, which justifies the use of *SSM* as a surrogate measure for *SM*.

Tabu-Search Algorithm:

The performance of the tabu-search algorithm is assessed via simulation. The results are summarized in Tables 5.9 and 5.10. Levels of machine routing are shown in columns and the rows list the levels of coefficient of variation and five initial solutions (SVPT, LEPT, SCV, SEPT and BS). The numbers in each cell report the average simulated *SM* values of the corresponding problem class and initial solution. The values in the parentheses are the number of times in which the corresponding initial solution yields the best *SSM* value. We observe that the average objective function values are close to each other and all initial solutions are competitive. In increasing the quality of the tabu-search algorithm, each initial solution method has its own contribution and none of them is superior, as in the case with no breakdowns.

Tables 5.11 and 5.12 present the percentage improvement in the simulated *SM* values (rather than estimated *SSM* values, which are actually used by the algorithms) for each seed schedule. We note that tabu-search improves the seed schedules' performance about 11% in average and generally, the improvement for the flow shop problems is more significant than the improvement for the job shop instances.

This can be explained with the intuition that flow shop problems are more challenging than job shop problems (Singer and Pinedo, 1998). This fact can also be observed by comparing Tables 5.2 and 5.4. Especially when the coefficient of variation is low, deviations from optimality are more for flow shop problems as compared to job shop instances. Thus, tabu-search has more room for improvement for flow shop problems.

Table 5.9. Summary of Results for Tabu-Search under Mild Breakdowns

		5x5			5x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	18421.39 (2)	14104.54 (7)	16243.631 (9)	23388.68 (4)	**17015.732 (6)**	**15694.892 (6)**
	LEPT	19421.57 (1)	15107.595 (4)	15980.561 (9)	23368.33 (2)	17812.333 (4)	16003.64 (5)
	SCV	19259.24 (4)	14354.925 (7)	16117.461 (8)	25138.53 (3)	17155.472 (3)	16474.112 (3)
	SEPT	19167.82 (4)	**14045.96 (5)**	15985.991 (8)	24752.9 (6)	17362.462 (6)	16711.88 (4)
	BS	**17017.81 (1)**	14287.829 (2)	**15335.73 (1)**	**21859.86 (1)**	17104.05 (0)	16443.39 (3)
CV2	**SVPT**	124270.7 (8)	119370.95 (7)	116422.86 (7)	184754.9 (2)	172621.1 (5)	157585.5 (7)
	LEPT	126820.5 (4)	118864.21 (2)	119438.15 (8)	201923.4 (2)	177921 (0)	157960.4 (7)
	SCV	129975.84 (5)	121541.38 (5)	117492.39 (7)	182985.1 (1)	177356.6 (2)	159998.3 (5)
	SEPT	125270.44 (8)	**115480.48 (5)**	**116247.28 (8)**	**177870.8 (6)**	**164881.8 (3)**	**155708.1 (8)**
	BS	**120049.24 (5)**	118875.93 (1)	116686.34 (8)	186165 (2)	169859.3 (4)	**155708.1 (8)**
CV3	**SVPT**	583335.7 (4)	606244.7 (7)	625739 (10)	871723.7 (2)	**917955.2 (4)**	**825981.1 (7)**
	LEPT	593459.4 (3)	621314.2 (6)	**623686 (9)**	910863.9 (1)	921765.5 (3)	835110.8 (7)
	SCV	619173.3 (0)	**598701.7 (3)**	625739 (10)	**858135.9 (1)**	948781.4 (3)	834215.7 (9)
	SEPT	595746.9 (4)	615792.2 (5)	624718 (9)	879600.1 (4)	928925.3 (5)	828945.5 (5)
	BS	**593008.9 (4)**	615463.6 (5)	625739 (10)	881972.2 (4)	919884.8 (3)	839346.6 (7)

		10x5			10x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	71106.21 (0)	63706.49 (0)	60105.01 (3)	86981.93 (1)	74011.3 (1)	66505.96 (2)
	LEPT	74239.34 (0)	65780.22 (1)	62697.36 (3)	89921.98 (1)	76916.48 (0)	69642.53 (2)
	SCV	72863.97 (1)	**62182.45 (1)**	**57753.83 (0)**	90250.93 (1)	77872.35 (2)	**62273.44 (2)**
	SEPT	72763.02 (1)	64951.79 (5)	61698.25 (2)	86597.19 (2)	78570.49 (6)	68988.91 (1)
	BS	**68964.98 (8)**	77622.61 (3)	64762.46 (2)	**84920.31 (5)**	**63771.12 (1)**	66003.36 (3)
CV2	**SVPT**	339990.8 (1)	316072.2 (2)	**297020.9 (4)**	500485.6 (2)	459197.6 (2)	472234.1 (2)
	LEPT	338506.1 (3)	322336.2 (2)	300505 (1)	**495987.5 (2)**	470119.3 (2)	**440614.2 (4)**
	SCV	325949 (1)	327757.6 (2)	322400.2 (2)	520262.3 (0)	468299.5 (2)	466828.2 (1)
	SEPT	324299.7 (2)	334940.9 (1)	302884.2 (1)	508169.1 (2)	466975.7 (3)	469477 (2)
	BS	**320091.4 (3)**	**300068.7 (3)**	317837.1 (2)	499843.9 (4)	**447772.9 (1)**	445341.9 (1)
CV3	**SVPT**	1839148 (2)	1782005 (4)	1788830 (2)	2568082 (1)	2521363 (1)	2442241 (4)
	LEPT	1902661 (3)	1813959 (2)	1738125 (0)	2618369 (1)	2463562 (7)	2507039 (1)
	SCV	1852832 (2)	**1640542 (1)**	1779277 (1)	2608712 (0)	2637421 (0)	**2384138 (2)**
	SEPT	**1837727 (3)**	1754564 (0)	**1693574 (6)**	**2500894 (3)**	**2457247 (2)**	2405625 (1)
	BS	1901984 (0)	1711237 (3)	1791261 (1)	2528729 (5)	2553765 (0)	2430622 (2)

Table 5.10. Summary of Results for Tabu-Search under Heavy Breakdowns

		5x5			5x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	**37172.12 (4)**	35007.29 (5)	38804.88 (8)	41150.84 (2)	37759.09 (4)	36382.52 (9)
	LEPT	37745.25 (0)	34097.97 (7)	38584.87 (9)	41069.43 (2)	37208.6 (3)	**35366.85 (4)**
	SCV	38024.05 (0)	35212.19 (3)	39083 (9)	43603.95 (2)	37626.62 (5)	36601.69 (4)
	SEPT	38024.21 (4)	**33690.82 (4)**	38404.36 (7)	**40339.39 (2)**	37926.82 (5)	36332.5 (6)
	BS	40145.73 (3)	36038.36 (0)	**36390.87 (0)**	52745.22 (2)	**35841.9 (0)**	40750.86 (0)
CV2	**SVPT**	**165399.8 (5)**	**167083.7 (8)**	**158256.4 (8)**	**218664.7 (3)**	**202837.5 (7)**	214556.4 (4)
	LEPT	178567.1 (1)	172292 (4)	161081.2 (6)	236503.6 (1)	216719 (1)	216420 (8)
	SCV	186597.3 (2)	175444.7 (2)	158236.6 (8)	224976.9 (0)	207517.6 (1)	217047.6 (6)
	SEPT	168519.3 (3)	168113.1 (3)	158679.5 (9)	222833.2 (5)	216643.1 (1)	212820.8 (9)
	BS	166468 (6)	174460.4 (0)	158614.5 (10)	219065 (2)	205308.5 (6)	**212239.1 (8)**
CV3	**SVPT**	**756235.5 (4)**	839599.2 (8)	729715.1 (10)	**1113801.5 (2)**	1040002.1 (4)	1197203.3 (7)
	LEPT	816598.9 (3)	838235.9 (4)	729663.8 (9)	1115376.5 (3)	1057882.7 (1)	**1179832.2 (7)**
	SCV	786169.8 (2)	832394.3 (4)	729715.1 (10)	1115448.4 (1)	1093406.6 (2)	1203427.2 (9)
	SEPT	782479.7 (5)	835020.5 (6)	**724717.2 (9)**	1172400.6 (4)	**1034097.1 (5)**	1184534.3 (6)
	BS	759420.6 (5)	**825894.3 (6)**	729715.1 (10)	1159365.8 (4)	1041889.9 (2)	1187727.2 (8)

		10x5			10x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	176737.9 (2)	155071.8 (2)	159891 (2)	231159.6 (2)	223841 (3)	207440 (1)
	LEPT	179932.2 (1)	156177.8 (1)	160312.2 (1)	227555.9 (1)	219157.8 (1)	**211550.1 (2)**
	SCV	182872.9 (2)	**150265.8 (2)**	**155253.9 (3)**	226534.8 (1)	**216535.3 (0)**	213443.1 (3)
	SEPT	183151 (2)	158229.7 (2)	161745.5 (2)	235154.1 (1)	225512.5 (3)	207610.1 (1)
	BS	**159492.1 (3)**	170974.2 (3)	163396.9 (2)	**220747.2 (5)**	222016.7 (3)	228728.3 (3)
CV2	**SVPT**	**492476.9 (2)**	439158.7 (1)	444269.4 (0)	711430.5 (2)	728781.2 (3)	725152.5 (1)
	LEPT	528641.8 (1)	**428578.7 (3)**	435769.3 (2)	731943.8 (2)	732758.5 (3)	**696551 (4)**
	SCV	502380.6 (3)	442861.9 (1)	437244.2 (4)	747750.2 (3)	753587 (2)	706391.6 (1)
	SEPT	498055.6 (3)	457876.7 (1)	435839.3 (2)	716117.7 (1)	**707410.3 (2)**	720040.6 (4)
	BS	512011.8 (1)	440436.9 (4)	**422045.9 (2)**	**708301.8 (2)**	718792.1 (0)	717035.8 (0)
CV3	**SVPT**	2204739 (1)	2067934 (3)	2289127 (4)	3651564 (0)	3548712 (0)	3502906 (0)
	LEPT	**2211235 (1)**	2075375 (3)	2316841 (2)	3470736 (1)	3521674 (1)	3519446 (3)
	SCV	2260545 (1)	**2053241 (2)**	2340920 (1)	**3426505 (1)**	3468209 (1)	3564149 (5)
	SEPT	2214541 (4)	2138645 (2)	**2287382 (3)**	3528729 (3)	**3360450 (7)**	3629918 (1)
	BS	2216375 (3)	2089639 (0)	2392343 (0)	3497881 (5)	3393413 (1)	**3431226 (1)**

Table 5.11. Contribution of Tabu-Search under Mild Breakdowns

		5x5			5x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	23.03%	17.17%	5.02%	20.34%	5.78%	3.05%
	LEPT	19.17%	14.68%	10.54%	17.84%	17.53%	19.01%
	SCV	18.02%	26.97%	6.51%	23.75%	20.67%	8.97%
	SEPT	14.69%	7.15%	12.06%	3.36%	16.49%	-1.24%
	BS	9.67%	8.07%	2.95%	11.76%	1.10%	-3.67%
CV2	**SVPT**	13.38%	12.01%	7.75%	8.01%	10.88%	3.90%
	LEPT	16.15%	18.99%	14.23%	6.56%	12.74%	6.69%
	SCV	22.22%	15.06%	14.85%	18.91%	10.97%	6.29%
	SEPT	3.45%	10.15%	12.59%	6.69%	17.59%	-1.67%
	BS	2.49%	4.16%	2.71%	2.38%	0.54%	1.18%
CV3	**SVPT**	12.88%	9.50%	4.95%	10.23%	10.37%	4.61%
	LEPT	21.84%	10.64%	11.33%	15.26%	17.94%	10.45%
	SCV	17.53%	21.86%	3.43%	20.33%	12.07%	10.43%
	SEPT	6.20%	4.34%	6.36%	3.31%	14.75%	0.25%
	BS	0.05%	0.27%	-4.46%	2.23%	-0.47%	0.45%

		10x5			10x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	24.59%	25.15%	4.97%	20.25%	12.92%	1.53%
	LEPT	12.15%	11.21%	15.52%	15.89%	3.59%	13.68%
	SCV	32.86%	27.02%	9.13%	22.33%	9.70%	14.70%
	SEPT	30.11%	13.01%	-2.00%	18.98%	5.14%	2.67%
	BS	6.08%	5.94%	-1.45%	1.79%	4.12%	3.98%
CV2	**SVPT**	14.78%	16.07%	12.52%	16.09%	12.25%	13.92%
	LEPT	13.57%	16.32%	21.27%	18.88%	5.72%	23.31%
	SCV	28.85%	21.57%	9.17%	13.09%	17.63%	22.24%
	SEPT	25.91%	11.05%	12.99%	4.84%	11.61%	12.03%
	BS	7.00%	9.71%	1.19%	-0.96%	6.61%	5.28%
CV3	**SVPT**	14.85%	9.95%	7.42%	9.30%	8.33%	7.68%
	LEPT	11.88%	13.61%	12.02%	13.34%	7.76%	11.05%
	SCV	21.45%	27.69%	12.38%	6.86%	8.76%	17.94%
	SEPT	19.77%	14.23%	11.43%	7.07%	8.42%	8.15%
	BS	4.85%	10.94%	1.52%	-0.18%	-1.70%	3.72%

Table 5.12. Contribution of Tabu-Search under Heavy Breakdowns

		5x5			5x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	25.04%	14.11%	10.22%	27.60%	18.19%	0.27%
	LEPT	23.64%	26.33%	8.94%	23.04%	22.11%	15.72%
	SCV	20.87%	28.62%	6.85%	36.05%	21.60%	8.39%
	SEPT	14.50%	8.02%	10.82%	15.86%	13.58%	1.71%
	BS	1.98%	-1.01%	10.03%	4.38%	-1.12%	0.93%
CV2	**SVPT**	15.00%	9.94%	5.44%	9.24%	21.32%	2.21%
	LEPT	16.41%	17.24%	14.44%	-0.15%	19.74%	9.44%
	SCV	15.70%	15.49%	12.63%	21.86%	20.97%	8.17%
	SEPT	5.93%	5.55%	8.27%	7.59%	22.91%	2.04%
	BS	5.15%	1.22%	3.73%	3.29%	7.69%	0.45%
CV3	**SVPT**	10.57%	4.36%	7.44%	9.37%	11.07%	1.44%
	LEPT	6.26%	16.09%	8.89%	16.04%	14.89%	7.02%
	SCV	14.31%	16.96%	10.74%	17.63%	12.98%	1.70%
	SEPT	3.77%	2.50%	12.72%	0.11%	20.59%	0.22%
	BS	2.24%	6.87%	-1.43%	1.31%	3.31%	-1.57%

		10x5			10x10		
		fixed	**semi**	**random**	**fixed**	**semi**	**random**
CV1	**SVPT**	23.64%	18.88%	6.23%	14.98%	14.24%	13.03%
	LEPT	4.90%	11.98%	16.77%	19.13%	4.17%	10.62%
	SCV	27.70%	23.01%	5.64%	26.60%	19.60%	13.71%
	SEPT	26.33%	13.33%	5.95%	14.34%	13.07%	5.50%
	BS	0.80%	12.26%	3.64%	1.15%	6.94%	-1.03%
CV2	**SVPT**	19.23%	16.17%	9.05%	16.62%	10.08%	7.86%
	LEPT	11.47%	22.50%	23.29%	18.29%	10.42%	14.34%
	SCV	30.15%	21.53%	22.64%	13.86%	14.52%	18.80%
	SEPT	25.45%	10.81%	10.36%	7.59%	20.18%	8.05%
	BS	0.12%	7.32%	8.63%	0.99%	3.38%	6.36%
CV3	**SVPT**	15.65%	19.32%	13.47%	4.42%	6.53%	14.12%
	LEPT	21.77%	22.14%	15.09%	10.05%	9.38%	14.95%
	SCV	22.16%	27.72%	9.38%	13.67%	17.28%	14.86%
	SEPT	20.20%	15.71%	9.83%	6.16%	11.56%	7.32%
	BS	8.19%	12.46%	-0.93%	-0.98%	7.23%	3.64%

5.9 Concluding Remarks

In this chapter, we study proactive scheduling in a job shop environment with random processing times and random machine breakdowns. We use total variance of the job completion times as the stability criterion. A surrogate stability measure is employed to generate stable schedules since calculating the stability measure analytically is impractical. The computational experiments indicate that there is a high positive correlation (> 0.97) between the defined stability measure and its surrogate. In this study, it is shown that minimizing even the surrogate stability measure is $\mathcal{NP}$-hard. We develop two branch-and-bound algorithms that optimize the surrogate stability measure in the class of active and non-delay schedules. We also develop two heuristics (a beam-search and a tabu-search algorithm) to handle large problems with machine breakdown/repair.

For exact algorithms, our computational experiments show that the impact of an increase in the number of jobs is more than the impact of an increase in the number of machines on the solution time. We observe that less computational time is needed to solve the instances with random machine routings than the instances with fixed or semi-random routings. It is also observed that as coefficient of variation increases the computational time also increase. We note that for large problems, it is practical to search the set of non-delay schedules rather than the larger set of active schedules to generate stable schedules.

Our computational experiments show that the proposed beam-search algorithm outperforms several dispatching rules (SVPT, LEPT, SCV and SEPT). When they are taken as seed schedules in the tabu-search algorithm, however, they yield schedules with close performances and they all are competitive. Hence, starting from multiple seeds is beneficial. We also note that tabu-search can potentially result in schedules with objective function values better than all active schedules in the case of no breakdowns, which includes the optimal solutions of the proposed branch-and-bound algorithms. We conclude that the proposed tabu-search algorithm is quite promising for generating stable schedules for large problems with random machine breakdowns.

We identify several further research directions. First, the proposed algorithms can be specialized to flow shop environments. Our computational experiments provide evidence to suspect that machine routings affect the solution quality. Algorithms that are customized for a flow shop environment may perform better than the general job shop algorithms developed in this chapter. Additionally, the job population in this study is fixed and all jobs are available at time zero. Including non-zero ready times and dynamic job arrivals will make the approach more applicable to real-life problems.

Second, both robustness and stability are important performance measures for the practitioners. Similar algorithms to generate robust job shop schedules can be developed. Moreover, a bicriterion algorithm that can handle both measures is of practical importance in the job shop environment. The relationship and the tradeoff between robustness and stability can also be analyzed like in Chapter 4.

Finally, different stability measures and better surrogates can also be developed. Even though our computational results indicate that there is a high positive correlation between the proposed stability measure and its surrogate, the possibility to employ simulation in order to estimate the stability performance of the schedules (contrasted to employing a surrogate measure) may be beneficial.

Chapter 6

Conclusion

In this thesis we study the machine scheduling in the face of random disruptions. We consider two sources of uncertainty: machine breakdowns and processing time variability. The information about these sources is modeled using cumulative distribution functions and formal probability theory is utilized to make inferences about the specific problems that are considered. We then draw advantage of these inferences to develop exact solution procedures where applicable. Several heuristics are also developed to improve the ability to handle the problems and to make real life applications possible.

In Chapter 3, we model uncertainty regarding job processing times and machine reliability with known probability distributions. We define several robustness and stability measures. This chapter contributes to the existing proactive scheduling literature in two ways. First, we identify the analytically tractable cases and we develop an exact algorithm to solve the common problem of minimizing the expected total tardiness using the insights gained while studying these cases. Second, for intractable cases, rather than taking an indirect approach by employing surrogate measures, we estimate the actual measures directly using simulation. The use of simulation in the existing studies may have been avoided because of its anticipated high computational burden. Our computational results, however, indicate that a beam-search algorithm that employs simulation as a global evaluation function is quite promising and requires reasonable computational times.

We can identify several further research directions. First, the proposed beam-search algorithm can be extended to more general multi-machine environments.

Additionally, the job population in this study is fixed and all jobs are available at time 0. Inclusion of non-zero ready times and dynamic job arrivals will make the approach more applicable to real-life problems.

Second, robustness can be measured from different points of view. For example, β-robustness can be studied. A β-robust schedule maximizes the probability of achieving a system performance less than or equal to a given threshold level T (Daniels and Carillo, 1997). The robustness in that sense for the due-date related performance measures can be investigated. Along the same lines, new, easy-to-calculate robustness or stability measures can be developed. There are other approaches in the literature that are used when dealing with uncertainty, including scenario planning and modeling with fuzzy numbers. We believe that such approaches could help alleviate the problems encountered in an analytical approach, such as the one taken in Chapter 3.

In Chapter 4, we study proactive scheduling in a single machine environment with random processing times. We use total expected flowtime and total variance of job completion times as the robustness and stability measures, respectively. A bicriteria approach to minimize both measures simultaneously is discussed. The proposed ε-constraint method, which generates the set of all Pareto optimal points, is more thorough than the common approach of combining both objective functions into a linear composite objective function. It is frequently used in multi criteria decision making studies in different fields, including machine scheduling. Three different versions of the ε-constraint method are investigated: the first one solves two instances of $\mathcal{NP}$-hard problems to obtain a Pareto optimal point whereas the second and the third ones solve only one such problem. The obtained point may be weak Pareto optimal in the second version. A dominance rule and three ways to formulate this rule are developed to get rid of some of weak Pareto points in this version.

Our computational experiments indicate that incorporating the dominance rule to the problem formulation at each iteration may in fact lower the number of weak Pareto points, especially in the presence of a negative correlation between the processing time mean and variance values. Our experiments, however, demonstrate that generating weak Pareto points and eliminating them is cheaper in terms of computational time than avoiding them. The computational results also show that the

presence of a negative correlation between processing time means and variances increase the total number of Pareto optimal points. Total number of Pareto points also increase as the number of jobs increase. Additionally, an increase in mean and variance ranges also causes a rapid increase in total number of points, which gives evidence to our suspect that the number of Pareto points may be pseudo polynomial in number of jobs. We also note that although a single iteration of the algorithm takes very little computational time, increasing problem sizes cause a rapid increase in total number of Pareto points and hence in total number of iterations required to generate the whole set. This suggests that being able to define the characteristics and shape of the trade-off curve using fewer Pareto points is of the essence. To that end, we propose the δ-grid search approach which generates a fixed number (set by the decision maker) of near-Pareto points.

Even though scheduling with more than one objective has been studied since 1980s, optimizing robustness and stability simultaneously in a proactive way is not thoroughly considered in the literature. The previous studies either preferred to include stability into the picture later in the reactive phase after an initial schedule is at hand or stability alone is optimized by inserting additional idle time into the schedules with the hope that the primary objective does not worsen a lot. The contribution of this chapter to the literature is that it provides a reliable method to consider both robustness and stability together, which is helpful to generate balanced schedules, especially if uncertainty is an inseparable part of the shop floor environment.

We can emphasize several areas to perform further research. First, the proposed approaches can be extended to other robustness and stability measures. Although multi criteria scheduling is not a new topic, most of the research effort is focused on earliness/tardiness problems or minimizing two regular performance measures at the same time. We believe that using the available toolbox of multi criteria techniques may help decision makers a great deal when coping with uncertainty. Second, the analysis can be extended to the more general shop floor environments such as shops with parallel machines, flow shops or job shops. Finally, algorithms that discover the characteristics of the trade off curve more cleverly may be developed. The brute force approach of generating the whole set of Pareto points may be impractical in terms of computational time requirements. Evolutionary meta heuristics are successfully being used in multicriteria decision making literature for this purpose.

Finally, we study proactive scheduling in a job shop environment with random processing times and random machine breakdowns in Chapter 5. We use total variance of the job completion times as the stability criterion. A surrogate stability measure is employed to generate stable schedules since calculating the stability measure analytically is impractical. The computational experiments indicate a high positive correlation (> 0.97) between the defined stability measure and its surrogate. In this study, it is shown that minimizing even the surrogate stability measure is $\mathcal{NP}$-hard. We develop two branch-and-bound algorithms that optimize the surrogate stability measure in the class of active and non-delay schedules. We also develop two heuristics (a beam-search and a tabu-search algorithm) to handle large problems with machine breakdown/repair.

For exact algorithms, our computational experiments show that the impact of an increase in the number of jobs is more than the impact of an increase in the number of machines on the solution time. We observe that less computational time is needed to solve the instances with random machine routings than the instances with fixed or semi-random routings. It is also observed that as coefficient of variation increases the computational time also increase. We note that for large problems, it is practical to search the set of non-delay schedules rather than the larger set of active schedules to generate stable schedules.

Our computational experiments show that the proposed beam-search algorithm outperforms several dispatching rules (SVPT, LEPT, SCV and SEPT). When they are taken as seed schedules in the tabu-search algorithm, however, they yield schedules with close performances and they all are competitive. Hence, starting from multiple seeds is beneficial. We also note that tabu-search can potentially result in schedules with objective function values better than all active schedules in the case of no breakdowns, which includes the optimal solutions of the proposed branch-and-bound algorithms. We conclude that the proposed tabu-search algorithm is quite promising for generating stable schedules for large problems with random machine breakdowns.

We again can point out several further research directions. First, the proposed algorithms can be specialized to flow shop environments. Our computational experiments provide evidence to suspect that machine routings affect the solution quality. Algorithms that are customized for a flow shop environment may perform

better than the general job shop algorithms developed in this paper. Additionally, the job population in this study is fixed and all jobs are available at time zero. Including non-zero ready times and dynamic job arrivals will make the approach more applicable to real-life problems.

Second, both robustness and stability are important performance measures for the practitioners. Similar algorithms to generate robust job shop schedules can be developed. Moreover, a bicriterion algorithm that can handle both measures is of practical importance. The relationship and the tradeoff between robustness and stability can also be analyzed.

Finally, different stability measures and better surrogates can also be developed. Even though our computational results indicate that there is a high positive correlation between the proposed stability measure and its surrogate, the possibility to employ simulation in order to estimate the stability performance of the schedules (contrasted to employing a surrogate measure) may be beneficial.

Bibliography

[1] Adiri I., J. Bruno, E. Frostig, and A. H. G. Rinnooy Kan (1989). Single machine flow-time scheduling with a single breakdown. *Acta Informatica 26(7),* 679-696.

[2] Adiri I., E. Frostig and A. H. G. Rinnooy Kan (1991). Scheduling on a single machine with a single breakdown to minimize stochastically the number of tardy jobs. *Naval Research Logistics 38(2),* 261-271.

[3] Akturk, M. S. and E. Gorgulu (1999). Match-up scheduling under a machine breakdown. *European Journal of Operational Research 112(1),* 81-97.

[4] Aytug H., M. A. Lawley, K. McKay, S. Mohan, and R. Uzsoy (2005). Executing production schedules in the face of uncertainties: a review and some future directions. *European Journal of Operational Research 161(1),* 86-110

[5] Chankong V. and Y. Haimes (1983). *Multiobjective Decision Making Theory and Methodology.* Elsevier Science, New York.

[6] Chu C. (1992). A branch and bound algorithm to minimize total tardiness with different release times. *Naval Research Logistics 39(2),* 265-283.

[7] Church, L. K., and R. Uzsoy (1992). Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing 5(3),* 153-163.

[8] Daniels, R. and P. Kouvelis (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science 41(2),* 363-376.

[9] Daniels, R. L. and J. E. Carillo (1997). β-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions 29(11),* 977-985.

[10] Della Croce, F., R. Tadei, P. Baracco, and A. Grosso (1998). A new decomposition approach for the single machine total tardiness scheduling problem. *Journal of the Operational Research Society 49(10),* 1101-1006.

[11] Du J. and T. Leung (1990). Minimizing total tardiness on one machine is NP-hard. *Operations Research 15(3),* 483-495.

[12] Evans G. W. (1984). An overview of techniques for solving multiobjective mathematical programs. *Management Science 30(11),* 1268-1282

[13] Fisher, M. L. (1976). A dual algorithm for the one machine scheduling problem. *Mathematical Programming 11(1),* 229-251.

[14] Giffler, B. and G. L. Thompson (1960). Algorithms for solving production-scheduling problems. *Operations Research 8(4),* 487-503.

[15] Goren, S. and I. Sabuncuoglu (2008). Robustness and stability measures for scheduling: single-machine environment. *IIE Transactions 40(1),* 66-83

[16] Herroelen, W. and E. Leus (2005). Project scheduling under uncertainty: survey and research potentials. *European Journal of Operational Research 165(2),* 289-306

[17] Hoogeveen H. (2005). Multicriteria scheduling. *European Journal of Operational Research 167(3),* 592-623.

[18] Kempf K., R. Uzsoy, S. Smith, and K. Gary (2000). Evaluation and comparison of production schedules. *Computers in Industry 42(2-3),* 203 - 220

[19] Kouvelis P. and G. Yu (1997). *Robust Discrete Optimization and Its Applications.* Kluwer Academic Publishers.

[20] Law, A.M. and W.D. Kelton (2000). *Simulation Modeling and Analysis*, 3rd ed., McGraw-Hill, Singapore.

[21] Leon, V. J., S. D. Wu, and R. H. Storer (1994). Robustness measures and robust scheduling for job shops. *IIE Transactions 26(5),* 32-43.

[22] Leung, J. Y.-T. and M. Pinedo (2004). A note on scheduling parallel machines subject to breakdown and repair. *Naval Research Logistics 51(1),* 60-71.

[23] Li, W., W.J. Braun, and Y.Q. Zhao (1998). Stochastic scheduling on a repairable machine with Erlang uptime distribution. *Advances in Applied Probability 30(4),* 1073-1088.

[24] Li, W. and K. D. Glazebrook (1998). On stochastic machine scheduling with general distributional assumptions. *European Journal of Operational Research 105(3),* 525-536.

[25] Mazzola, J. B. and A. W. Neebe (1986). Resource-constrained assignment Scheduling. *Operations Research 34(4),* 560 – 572

[26] Mehta, S. V. and R. Uzsoy (1998). Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation 14(3),* 365-378.

[27] O'Donovan, R., R. Uzsoy, and K. N. McKay (1999). Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research 37(18),* 4217-4233.

[28] Özlen, M. and M. Azizoğlu (2009). Multi-objective integer programming: a general approach for generating all non-dominated solutions. *European Journal of Operational Research 199(1),* 25-35.

[29] Pinedo, M. (2002). *Scheduling Theory, Algorithms, and Systems*. Upper Saddle River, New Jersey 07458: Prentice Hall.

[30] Ross, S. M. (1983). *Stochastic Processes*. John Wiley & Sons.

[31] Ross, S. M. (1993). *Introduction to Probability Models*. Academic Press, Inc.

[32] Nelson, R. T., C.A. Holloway, and R. M. Wong (1977). Centralized scheduling and priority implementation heuristics for a dynamic job shop model. *IIE Transactions 9(1),* 95-102.

[33] Sabuncuoglu I. and S. Goren (2009). Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability

research. *International Journal of Computer Integrated Manufacturing 22(2)*, 138-157.

[34] Sabuncuoglu I. and S. Karabuk (1998). A beam search based algorithm and evaluation of scheduling approaches. *IIE Transactions 30(2)*, 179-191.

[35] Sabuncuoglu I. and S. Karabuk (1999). Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems 18(4)*, 268-283.

[36] Sabuncuoglu I. and M. Bayiz (1999). Job shop scheduling with beam search. *European Journal of Operational Research 118(2)*, 390-412.

[37] Sabuncuoglu I. and M. Bayiz (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research 126(3)*, 567-586.

[38] Singer M. and M. Pinedo (1998). A computational study of breanch and bound techniques for minimizing the total weighted tardiness in job shops. *IIE Transactions 30(2)*, 109-118.

[39] Sotskov Y., N. Y. Sotskova, and F. Werner (1997). Stability of an optimal schedule in a job shop, *Omega: the International Journal of Management Science, 25(4)*, 397-414.

[40] T'kindt, V. and J.-C. Billaut (2002). *Multicriteria Scheduling: Theory, Models, and Algorithms*. Springer, Berlin.

[41] Vieira, G.E., J.W. Herrmann, and E. Lin (2003). Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of Scheduling 6(1)*, 39-62.

[42] Vepsalainen, A. P. J., and T. E. Morton (1987). Priority rules for job shops with weighted tardiness costs. *Management Science 33(8)*, 1035-1047.

[43] Wu, S. D., E. Byeon, and R. H. Storer (1999). A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research 47(1)*, 113-124

[44] Wu, S. D., R. H. Storer, and P. Chang (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers Ops Res. 20(1)*, 1-14.