Winter 2007

Exact and Heuristic Algorithms for the Job Shop Scheduling Problem with Earliness and Tardiness Over a Common Due Date

Leonardo Bedoya-Valencia
Old Dominion University

EXACT AND HEURISTIC ALGORITHMS FOR THE JOB SHOP SCHEDULING PROBLEM WITH EARLINESS AND TARDINESS OVER A COMMON DUE DATE

by

Leonardo Bedoya-Valencia
M.Sc., Universidad Nacional de Colombia
Ingeniero Industrial, Universidad Nacional de Colombia

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ENGINEERING MANAGEMENT AND SYSTEMS ENGINEERING

OLD DOMINION UNIVERSITY
December 2007

Approved by:

Ghaith Rabadi (Director)

Andrés Sousa-Poza (Member)

Resit Unal (Member)

Ali Ardalan (Member)

ABSTRACT

EXACT AND HEURISTIC METHODS FOR THE JOB SHOP SCHEDULING PROBLEM WITH EARLINESS AND TARDINESS OVER A COMMON DUE DATE

Leonardo Bedoya-Valencia
Old Dominion University, 2007
Director: Ghaith Rabadi, Ph.D.

Scheduling has turned out to be a fundamental activity for both production and service organizations. As competitive markets emerge, Just-In-Time (JIT) production has obtained more importance as a way of rapidly responding to continuously changing market forces. Due to their realistic assumptions, job shop production environments have gained much research effort among scheduling researchers. This research develops exact and heuristic methods and algorithms to solve the job shop scheduling problem when the objective is to minimize both earliness and tardiness costs over a common due date. The objective function of minimizing earliness and tardiness costs captures the essence of the JIT approach in job shops. A dynamic programming procedure is developed to solve smaller instances of the problem, and a Multi-Agent Systems approach is developed and implemented to solve the problem for larger instances since this problem is known to be NP-Hard in a strong sense. A combinational auction-based approach using a Mixed-Integer Linear Programming (MILP) model to construct and evaluate the bids is proposed. The results showed that the proposed combinational auction-based algorithm is able to find optimal solutions for problems that are balanced in processing times across machines. A price discrimination process is successfully implemented to deal with unbalanced problems. The exact and heuristic procedures developed in this research are

the first steps to create a structured approach to handle this problem and as a result, a set of benchmark problems will be available to the scheduling research community.

To Sofia and Yaneth, whose love keeps me going, and to my family for their unconditional support

ACKNOWLEDGMENTS

Many people have contributed both professionally and personally to the completion of this dissertation. In particular, I would like to thank my advisor, Dr. Ghaith Rabadi, for his guidance, support, and patience during the time I was trying to finish this dissertation. Also, I would like to thank the members of my Doctoral committee Dr. Resit Unal, Dr. Andrés Sousa-Poza, and Dr. Ali Ardalan.

I gratefully acknowledge the support of the Engineering Management and Systems Engineering Department (EMSE), especially Dr. Charles Keating who has always believed not only in me, but also in my wife.

I would also like to thank the Faculty and the Staff of the EMSE department for their support and collaboration, especially Kim Sibson who took time from an important stage in her life to go through my dissertation.

I am grateful to my parents, my brothers, my sisters, and my nephew. Even if they are not here, they have been always with me. Finally, I would like to thank my wife Yaneth and my daughter Sofia for making me the happiest man ever.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I.

INTRODUCTION

Business organizations produce goods and/or provide services, and even though their goals and products are quite different, their functions and ways of operation are quite similar. Production/operation is one of the essential functions of virtually every business organization, and it overlaps with other functions, such as finance and marketing.

Scheduling has to interface with the productions/operations' basic functions, ranging from the production planning, which handles medium to long-term decisions, to shop floor control that handles short-term decisions. As a result, scheduling has become a fundamental activity for organizations (Pinedo 2002)[1].

Since the early 1970s, Just-in-time (JIT) management philosophy has been applied in manufacturing. JIT involves having the right items, in the right quality and quantity, at the right place and at the right time. Cheng and Podolsky (1996) reported that the proper use of JIT has increased quality, productivity, efficiency and has reduced costs and waste. In this sense, productions/operations functions are the heart of the JIT philosophy, and they focus on elements such as plants, equipment, and production planning and control. JIT orients the production planning to the customer as the main performance target by using "pull" rather than "push" planning and control activities.

Viewed as an operational activity, scheduling determines the sequence of jobs or tasks on a particular machine or production line. It also determines the human resources

[1] References in this dissertation follow ACS style by author name and date.

and materials required to perform such tasks. In order to carry out this activity, some basic scheduling functions involving timing and allocation of the necessary resources, as well as sequencing and control of the jobs or tasks, need to be defined (Pinedo 2002).

By its very nature, scheduling in real environments very often considers multicriteria objectives, which involve time as well as cost related criteria (T'kindt and Billaut 2002). One of the classical objectives in scheduling is linked to due dates, which focus on meeting customers' delivery dates. As the term indicates, JIT is intended to avoid both earliness and tardiness, where the objective is to find a sequence of tasks such that they are completed as close as possible to their due dates. In addition to the cost of maintaining inventories, earliness could count for the amount of spoilage and the high investment cost(s) for special storage facilities required to keep perishable goods such as food and chemical products. On the other hand, tardiness is the most common measure to assess how well customer due dates are met.

Due dates can be determined as a result of a choice made by the decision maker or negotiation between the decision maker and the customer. Therefore, at least one criterion related to the tardiness of jobs and one criterion related to their earliness must be considered in the objective function, which turns the JIT scheduling problems into multicriteria optimization problems. Besides their applicability to some real situations, these problems are interesting because, in general, it is very difficult to find a schedule in which all jobs will be completed on time, and therefore, the decision maker must face a trade off between earliness and tardiness.

The central purpose of this research is to develop both exact and heuristic algorithms to find optimal or near optimal solutions for this bi-objective problem. The

exact algorithms are developed using and extending some of the existing properties of simpler versions of the problem in order to reach optimal solutions in polynomial or pseudo-polynomial time. Heuristic algorithms are developed and used when it is not possible to use exact algorithms, especially for large problems. General-purpose methods are considered in this research by developing a Multi-Agent System to find optimal or near optimal solutions in polynomial time.

Problem Statement

The problem addressed in this research is the job shop scheduling problem (JSSP), where a set of n available jobs must be scheduled on m machines. In the most general case, each job consists of m operations -on each machine. Each job i may have a different route, processing time p_{ij} on machine j, and a due date d_i. Because of its difficulty to be solved, the JSSP has been a challenge for researchers in the Operations Research area for a few decades (Conway et al 1967). Besides, in a job shop environment, the quantities made of one product are small, created typically according to specific customer requirements and, as a result, a wide variety of products can be produced. These features bring the job shop environment close to practical real scheduling problems.

In this research, the objective in the JSSP is to minimize both earliness and tardiness for all jobs. Let C_i, E_i and T_i represent the completion time, earliness, and tardiness of job i respectively, E_i and T_i can be defined as:

$$E_i = Max(0,\ d_i - C_i) = (d_i - C_i)+ \qquad \textbf{Equation 1}$$

$$T_i = Max(0,\ C_i - d_i) = (C_i - d_i)+ \qquad \textbf{Equation 2}$$

where $(.)^+$ represents the positive difference from the due date. Associated with each job there is an earliness penalty $\alpha_i > 0$ and a tardiness penalty $\beta_i > 0$ per time unit. Assuming that the penalty functions are linear, the basic earliness and tardiness (E/T) objective function for a schedule S can be written as $f(S)$ as follows:

$$Min \quad f(S) = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i) \qquad \textbf{Equation 3}$$

The JSSP E/T problem addressed in this research is the one with a common due date (CDD) where all jobs have the same due date (i.e. $d_i = d \; \forall i = 1, \ldots, n$). CDD becomes important when a set of components must be assembled into a finished product, or when several jobs must be shipped together to a certain customer. In a JIT environment, these jobs should be finished as close to the CDD as possible. An early job completion results in inventory and handling costs, and a tardy job completion results in customer penalties. As can be seen from Figure 1, each job in a job shop has its own route that defines that sequence of operations on the machines.

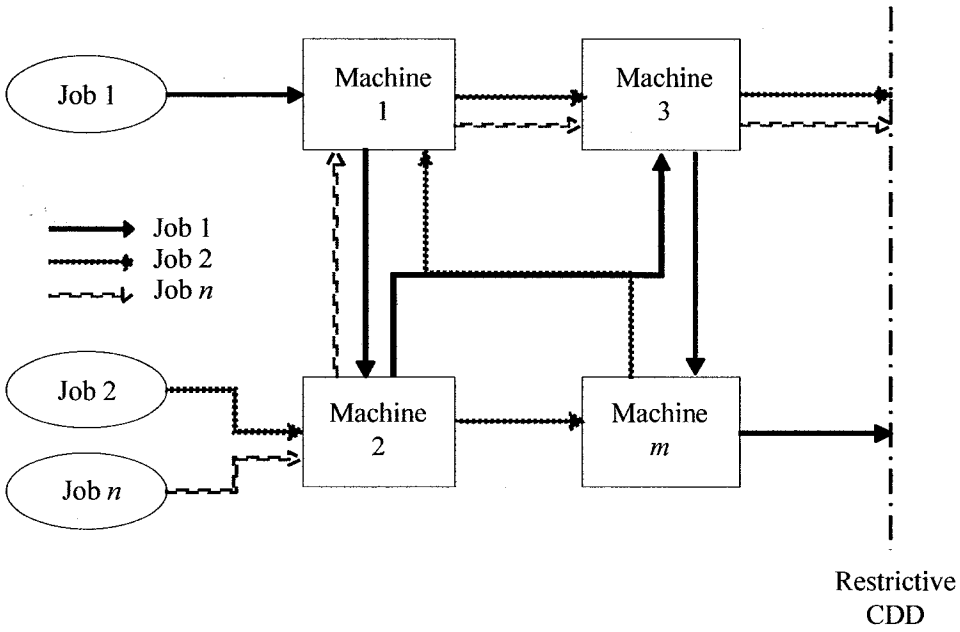


Figure 1 The Job Shop Scheduling Problem Considering Earliness and Tardiness over a Common Due Date (JSSP E/T CDD).

The CDD considered here is *restricted*, which means that it could be small enough to restrict the scheduling decision and it has influence on the optimal sequence. The restricted version of the problem is much harder than the unrestricted version (Lauff and Werner 2004a). As Baker (1997) stated, for the single machine E/T problem over a

CDD, it would be desirable to construct a schedule in which half of the jobs are completed before the CDD. If the CDD was too tight, then not enough jobs would be scheduled before the CDD, as they cannot start before time zero. In this case, this problem is known as *Restricted* as shown in Figure 2; otherwise, when the CDD is not too tight, it is known as the *Unrestricted Case*. The latter case can be solved by using polynomial algorithms for both the single machine and the JSSP E/T CDD (Lauff and Werner 2004a).

Formally, there is a quantitative procedure to define whether a CDD is restricted or unrestricted for the single machine problem, which is explained in detail in the methodological approach section. In this research, such a procedure will be extended to the two machines JSSP E/T CDD.

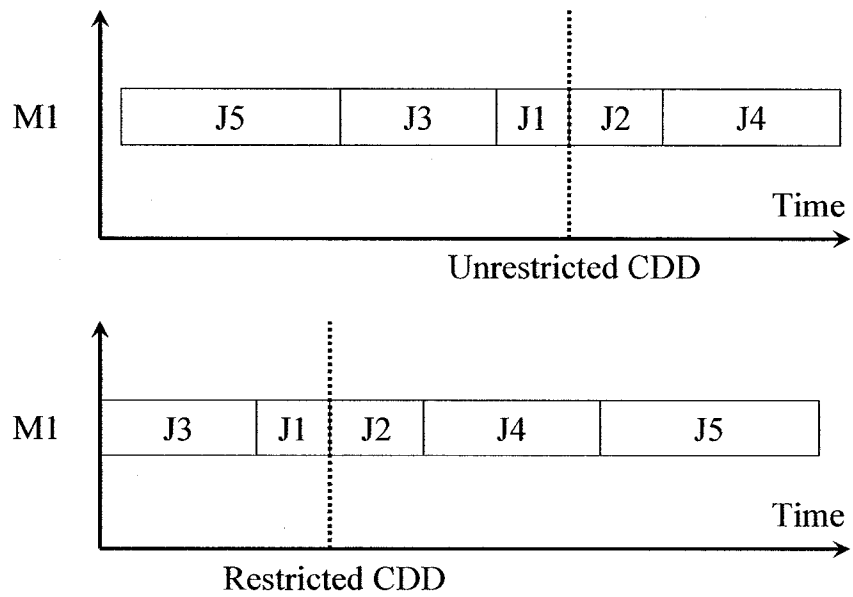


Figure 2 Unrestricted and Restricted Case of the Common Due Date (CDD).

The performance criterion used in this research (earliness and tardiness) is known to be a *non-regular* one. A performance criterion is considered regular if the objective function to be minimized increases as the completion times of the jobs increase (Pinedo 2002).

The search process aimed to look for optimal solutions for regular measures of performance has to be carried out in a defined search space. However, there are some scheduling problems, like the ones with non-regular measures of performance, including earliness and tardiness, where optimal solutions could be found in a larger search space. Figure 3 shows a Venn diagram of the search space for both regular and non-regular measures of performance.

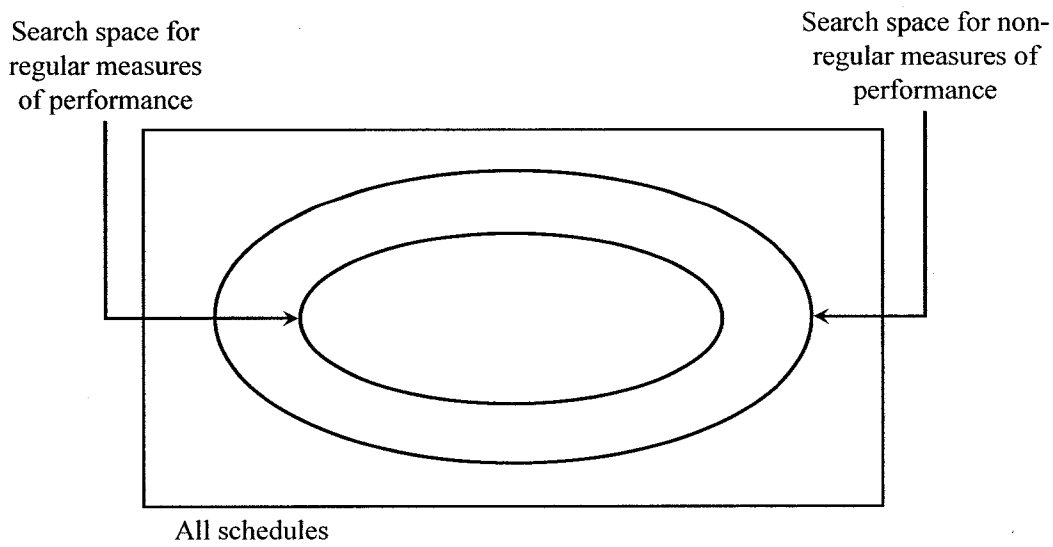


Figure 3 Venn Diagram of Classes of Schedules for JSSP.

Complexity of the Problem

A scheduling problem can be described by a triplet $\alpha / \beta / \gamma$ (Pinedo 1995) where α describes the machine environment and contains a single entry, β provides details of processing characteristics and constraints and can contain no entries, a single entry, or multiple entries, and finally, γ contains the objective to be minimized and usually only has a single entry. This description and the complexity hierarchy proposed by Pinedo (2002) are used to establish the complexity of the JSSP E/T CDD.

The scheduling problem $J_m // C_{max}$ is known to be NP-Hard (Garey et al, 1976), where J_m refers to a Job Shop environment and C_{max} is the objective of minimizing the makespan (i.e., the time needed to complete all jobs). This problem is NP-hard even if the number of machines is greater than or equal to two. Though this problem deals with a regular measure of performance, simpler problems dealing with non-regular measures of performance are also known to be NP-Hard, such as the single machine E/T scheduling problem over a restricted CDD (Hall et al, 1991).

By using a complexity hierarchy of objective functions Pinedo (2002), regular measures of performance related to tardiness and lateness are more complex than the one related to the makespan (C_{max}). Therefore, as a conjecture, it is possible to state that the scheduling problem $J_m / d_j = restricted\ CDD / \Sigma(\alpha_j E_j + \beta_j T_j)$ is NP-Hard.

Given that this problem is NP-Hard, heuristic approaches might be used to find a good, near-optimal solution. However, properties of some particular instances of the problem that could be exploited to develop polynomial or pseudo-polynomial algorithms to find optimal solutions will be proposed in the methodological approach section.

Methodological Approach

Figure 4 shows the general framework for the proposed research method to deal with the problem at hand. Based on both real production environment, and scheduling theory and concepts, a problem has been stated. Generally speaking, most of the problems to be solved in scheduling are a branch of optimization and are classified as NP-Hard. That is, there are no efficient algorithms that can find optimal solutions in a polynomial or pseudo-polynomial time to solve a NP-Hard problem. Instead, for a small number of optimization problems, there are exponential-based algorithms; although not considered efficient, they can solve these small problems, which are labeled as "well-solved" problems (Garey and Johnson 1979).

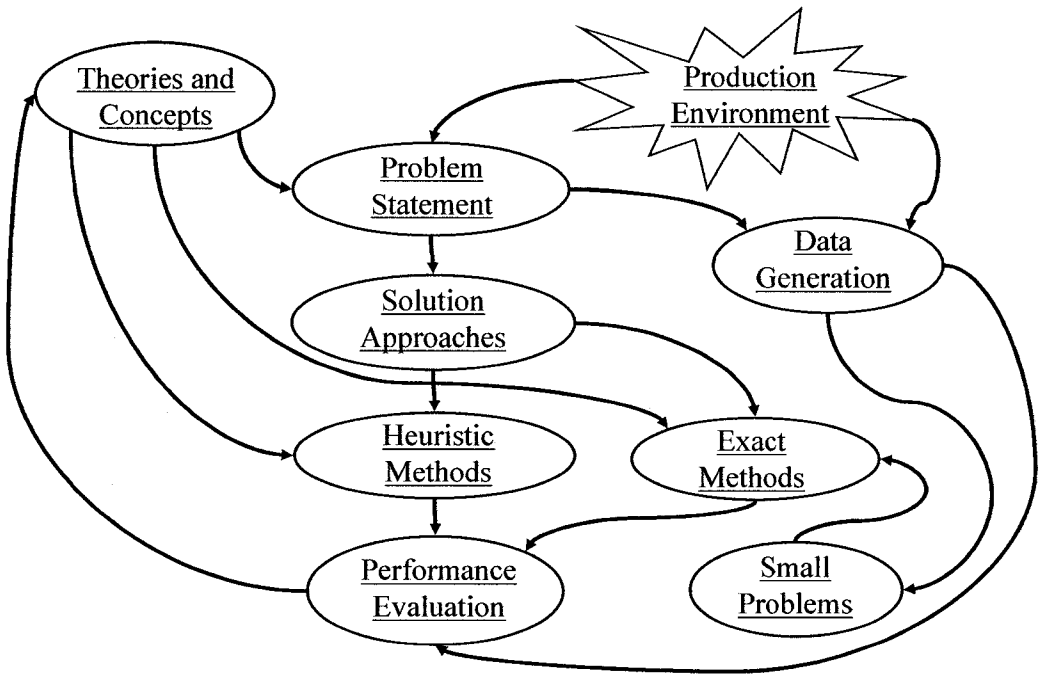


Figure 4 General Framework for the Proposed Research Method.

The most efficient method to find optimal solutions for scheduling problems is to intensively exploit properties and features of problems to be solved. The shortest processing time (SPT) rule to find an optimal solution for a single machine problem considering the mean flow time, and the earliest due date (EDD) rule to find an optimal solution for a single machine problem considering the maximum lateness when all jobs are available at time zero are among the examples of this approach (Baker 1997). According to Garey and Johnson (1979), some algorithms can capture important properties of the problems, and based on these properties, algorithms can be refined in order to lead to better methods to solve the problem. The main objective of this research is to study the basic properties of the JSSP E/T with a restricted CDD to incorporate in an efficient algorithm (polynomial, pseudo-polynomial or exponential) to solve the problem. A set of properties, already applied to some problems, will be generalized to a bigger set. Some of these properties have been derived for a less complicated machine-scheduling environment (single machine). In addition, some of the properties required to improve the algorithms are not derived from the single machine environment and need to be developed by using an inductive approach. Based on the patterns observed in optimal solution for small problems, optimal properties will be defined and proven for the more general JSSP E/T with m machines, n jobs, and a restricted CDD. After :Chapter II: Literature Review", properties of both the two-machine and multi-machine job shop scheduling problems over a CDD will be derived, in addition, exact and heuristic methods will be developed.

CHAPTER II

LITERATURE REVIEW

The literature review is structured in the following manner. Initially, papers dealing with the general properties of the earliness and tardiness (E/T) problems are reviewed. Exact methods to find optimal solutions for simpler problems, and their complexity are especially analyzed. Next, heuristic methods for similar problems are studied, emphasizing those related to Multi Agent Systems (MAS). Finally, and based on the review, a final statement related to the research gap for the proposed problem will be identified.

E/T Research on Optimal Properties and Solutions

Many of the published scheduling papers dealing with earliness and tardiness addressed the single-machine E/T problem. Baker and Scudder (1990) published a comprehensive state-of-the-art review for different variations of the E/T problem, including the problem with a CDD for all jobs. Gordon et al. (2002) have recently reviewed the literature of the E/T problem with CDD where the focus of their review was mainly on single and parallel machine scheduling problems as there is little research on open, flow and job shop E/T problems. Similarly, Lauff and Werner (2004a, 2004b) confirmed that there are only a few papers dealing with multi-stage systems involving earliness and tardiness problems with CDD.

Kanet (1981) developed an algorithm to find an optimal solution for the single machine E/T problem with an unrestricted CDD. In his work, some properties of optimal solutions were stated and proved and then used to construct a polynomial algorithm. Sundararaghavan and Ahmed (1984) developed a heuristic algorithm for the same problem, but with an arbitrary (restricted or unrestricted) CDD. They used some of the properties defined for the unrestricted case by Kanet (1981) when the CDD is small enough to constrain the schedule. Bagchi et al. (1986) extended Kanet's idea of an unrestricted CDD and developed an exact algorithm to generate alternate optimal solutions. They also developed an implicit enumeration procedure for the restricted case in a single machine environment. Raghavachari (1986) extended the V-shape property of optimal schedules established by Kanet (1981) to any CDD. Later, Szwarc (1989) studied a variation of this problem considering a fixed starting time for the first job in the schedule and a restricted CDD. He developed a Branch and Bound (B&B) procedure to find optimal solutions for problems with up to 25 jobs.

Hoogeven and Van de Velde (1991) developed a dynamic programming algorithm to solve the single machine scheduling problem considering a CDD and a positive weight for each job. In their work, they did not define the nature of the CDD (restricted or unrestricted). However, they found out that the problem with equal processing times for all the jobs and the problem with equal weight to processing time rates are polynomially solvable cases. Also, Hall and Posner (1991) developed a dynamic programming algorithm for the single machine problem considering an unrestricted CDD and different weights for the jobs. Hall et al. (1991) constructed an exact algorithm based on dynamic programming to find an optimal solution for the single

machine unweighted E/T problem with a restricted CDD. Rabadi et al (2004) developed a B&B procedure to find optimal solutions for single machine problems with an unrestricted CDD and considering sequence-dependent setup times.

As for multi-machine environments, Emmons (1987) developed an algorithm that is able to solve scheduling problems considering identical parallel machines when all jobs have a CDD, and when earliness and tardiness have different cost rates. His algorithm finds optimal solutions for problems where the number of jobs is less than or equal to four times the number of machines and finds good solutions in the rest of the cases. Federgruen and Mosheiov (1996) studied the identical parallel machines scheduling problem considering an unrestricted CDD and non-decreasing convex earliness and tardiness cost functions. They developed a lower bound for the cost, as well as a heuristic procedure to solve the problem. The heuristic was also generalized to include problems with a restricted CDD and general asymmetric, and possibly non-convex, earliness and tardiness cost functions. Bank and Werner (2001) studied the unrelated parallel machine scheduling problem with release dates and a CDD. The objective was to minimize the weighted sum of linear earliness and tardiness penalties. They derived some structural properties and used them to develop approximate constructive and iterative heuristic algorithms to solve the problem. Sun and Wang (2003) studied the problem of scheduling n jobs with a CDD and proportional earliness and tardiness penalties on m identical parallel machines. They showed that the problem is NP-Hard and proposed a dynamic programming algorithm to solve it. They also proposed two heuristics to deal with the problem and analyzed their worst-case error bounds.

Sarper (1995) developed a mixed integer linear programming formulation for the two-machine Flow Shop Scheduling Problems (FSSP) considering the unweighted earliness and tardiness cost over a CDD. In his approach, he used an arbitrary CDD and developed three heuristic procedures to solve the problem, which he compared with optimal solutions for problems. Sung and Min (2001) studied the two-machine FSSP with batch processing machines and a CDD. They defined some properties of three special cases and developed exact methods to find optimal solutions. Although they did not explicitly say that the CDD was unrestricted, they assumed that the CDD is greater than or equal to the time required for processing all jobs on the first machine. Recently, Gupta et al. (2004) defined some properties of optimal schedules for the two-machine FSSP with non-regular criteria (earliness and tardiness). Also, they developed lower and upper bounds, derived dominance criteria, and proposed an enumerative algorithm for finding an optimal schedule. Finally, Lauff and Werner (2004c) developed heuristic algorithms, both constructive and enumerative, to solve the two-machine FSSP with a given CDD considering asymmetric linear and quadratic penalty functions. Their algorithms were based on some structural properties of the problem. So far, there is no reported research on the JSSP considering E/T over a CDD.

Heuristic Methods for the E/T Problem

Feldmann and Biskup (2003) developed three meta-heuristic approaches to solve the single machine scheduling problem considering weighted earliness and tardiness penalties over a restricted CDD. Hino et al. (2005) developed a heuristic exploiting some

of the properties of the single machine problem considering earliness and tardiness penalties with a CDD; then they used some meta-heuristics and hybrid meta-heuristics to improve the solution. Rabadi et al. (2007) introduced a constructive heuristic for the single machine E/T with unrestricted CDD and sequence dependent-setup times and compared the heuristic's performance to a simulated annealing algorithm for the same problem. As mentioned earlier, Sarper (1995) worked on three heuristics to solve the FSSP with two machines and a CDD. Finally, Zegordi et al. (1995) applied simulated annealing to the FSSP considering early/tardy costs. Although each job has its own due date, the objective function considered in their work is still a non-regular measure of performance. Table 1 shows the literature review summary.

Table 1 Literature Review about the Problem Definition.

Reference	*Environment*	*Constraints*	*Method*	
			Heuristic	*Exact*
Kanet (1981)	Single machine	Unrestricted CDD		X
Sundararaghavan and Ahmed (1984)	Single machine	Arbitrary CDD	X	
Bagchi et al. (1986)	Single machine	Unrestricted CDD		X
Raghavachari (1986)	Single machine	Arbitrary CDD		X
Szwarc (1989)	Single machine	Fixed starting time, Restricted CDD		X
Hoogeven and Van de Velde (1991)	Single machine	Arbitrary CDD, Different weight for each job		X
Hall and Posner (1991)	Single machine	Arbitrary CDD, Different weight for each job		X
Hall et al. (1991)	Single machine	Restricted CDD, unweighted case		X
Feldmann and Biskup (2003)	Single machine	Restricted CDD, weighted case	X	

Reference	*Environment*	*Constraints*	*Method*	
			Heuristic	*Exact*
Rabadi et al (2004)	Single machine	Unrestricted CDD, unweighted case, setup times		X
Rabadi et al (2007)	Single machine	Unrestricted CDD, unweighted case, setup times	X	
Hino et al. (2005)	Single machine	Arbitrary CDD, weighted case		
Emmons (1987)	Identical Parallel Machines	Arbitrary CDD, early and tardy costs different	X	X
Federgruen and Mosheiov (1996)	Identical Parallel Machines	Unrestricted and restricted CDD, weighted case	X	
Bank and Werner (2001)	Unrelated Parallel Machines	Release dates, Arbitrary CDD	X	
Sun and Wang (2003)	Identical Parallel Machines	Arbitrary CDD, weighted case	X	X
Sarper (1995)	Two-machine Flow Shop Problem	Arbitrary CDD, unweighted case	X	X
Zegordi et al. (1995)	Flow Shop Problem	Multiple due date	X	
Sung and Min (2001)	Two-machine Flow Shop Problem	Unrestricted CDD, unweighted case, Batching possibility		X
Gupta et al. (2004)	Two-machine Flow Shop Problem	Arbitrary CDD, weighted case	X	
Lauff and Werner (2004c)	Two-machine Flow Shop Problem	Arbitrary CDD, linear and quadratic cost functions	X	

Among the more recent approaches for solving large problems is the use of Multi-Agent Systems (MAS), which seems to have good potential for solving complex problems.

Lin and Solberg (1992) designed a multi-agent based approach for shop floor control and scheduling, which was a market-like model for the control strategies where parts (jobs) and resource agents negotiated in a heterarchical environment (just one level of hierarchy as shown in Figure 9). Wellner and Dilger (1999) developed a negotiation control strategy where two types of agents were created: job and resource agents, which used a quasi heterarchical control process to minimize the makespan for a JSSP. Fabiunke and Kock (2000) used a sequencing control strategy by considering each operation in a JSSP as an agent with a heterarchical control process to minimize the makespan. Dang and Frankovic (2002) developed a negotiation control strategy and a heterarchical control process with jobs as agents to solve a flexible JSSP. Dewan and Joshi (2002) created a bidding control strategy to solve a dynamic JSSP where they considered machines as "auctioneer" agents and jobs as "bidder" agents in a heterarchical environment. Macchiaroli and Riemma (2002) proposed a multi-agent-based approach similar to the one proposed by Lin and Solberg (1992). In their research, they included the cooperation control strategy in order to reach a global optimal performance.

A few other papers have also addressed the total tardiness problem such as Biskup and Simons (1999) who developed a negotiation scheme with a heterarchical control process to solve the dynamic total tardiness problem in a job shop environment. They created different negotiation schemes using game theory. Kutanoglu and Wu (1999) developed a bidding control strategy and a heterarchical control process to minimize the tardiness for the JSSP. They used a combinational auction mechanism and a Lagrangean relaxation to efficiently allocate resources where the jobs were considered as agents in a heterarchical environment. Sabuncuoglu and Toptal (1999b, 1999c) developed a bidding

and cooperation control strategy to solve the JSSP considering setup times. They used a quasi-heterarchical control process with tardiness-related measures of performance. Also, with regard to the tardiness problem, Aydin and Oztemel (2000) developed an agent-based approach where an agent learns from both past data and the current state of the system, and then dispatches jobs in a dynamic job shop environment. Wu and Weng (2005) created a multi-agent approach to solve the flexible JSSP considering earliness and tardiness as the measure of performance. In their work, jobs and machines were considered as agents and bidding was used as a control strategy in a heterarchical fashion.

Combinational auctions, a control strategy that has been developed recently, has gained a lot of attention for solving complex resource allocation problems. This type of auction has the advantage of allowing bidders to express their synergistic values by submitting bids for combinations of assets. This structure fits very well into the multi-machine scheduling environment where multiple resources need to be allocated among different bidders.

Rothkopf et al (1998) discussed different applications of combinational auctions and identified different structures for combinational bidding where a computational implementation is feasible. More recently, Kutanoglu and Wu (1999) used a combinational auction to solve the JSSP involving total tardiness as measure of performance. Reeves et al (2005) explored different bidding strategies specifically for scheduling problems involving multiple resources and pointed out that most of the literature on auctions theory deals with a single resource. Consequently, using combinational auctions as a control strategy in the JSSP with a non-regular measure of

performance will add to the body of the knowledge on this topic. Table 2 shows the literature review with respect to different multi-agent approaches.

Table 2 Literature Review about Multi Agent Approaches.

Reference	*Environment*	*Control*	*Mechanism*	*Agents*		*Measure of Performance*
				Jobs	*Machines*	
Lin and Solberg (1992)	Job/Open Shop	Heterarchical	Negotiation	X	X	Regular
Wellner and Dilger (1999)	Job Shop	Quasi-heterarchical	Negotiation	X	X	Regular
Fabiunke and Kock (2000)	Job Shop	Heterarchical	Sequencing	X		Regular
Dang and Frankovic (2002)	Flexible Job Shop	Heterarchical	Negotiation	X	X	Regular
Dewan and Joshi (2002)	Job Shop	Heterarchical	Bidding	X	X	Regular
Macchiaroli and Riemma (2002)	Job/Open Shop	Heterarchical	Negotiation and Cooperation	X	X	Regular
Biskup and Simons (1999)	Dynamic Job Shop	Heterarchical	Negotiation	X	X	Regular
Kutanoglu and Wu (1999)	Job Shop	Heterarchical	Bidding	X	X	Regular
Sabuncuoglu and Toptal (1999b, 1999c)	Job Shop with setup times	Quasi-heterarchical	Bidding and Cooperation	X	X	Regular
Aydin and Oztemel (2000)	Dynamic Job Shop	Quasi-heterarchical	Sequencing			Regular
Wu and Weng (2005)	Flexible Job Shop	Heterarchical	Bidding	X	X	Non-Regular
Reeves et al (2005)	Single machine	Heterarchical	Bidding	X		Regular

Most of the research involving JSSP and heuristic methods has dealt with regular measures of performance, mainly the makespan. So far, we are unaware of any research

addressing the JSSP E/T with a restricted CDD. Due to the clear lack of research on the JSSP E/T with a restricted CDD, the development of both exact and heuristic methods would be of great importance and will represent a clear contribution to the body of knowledge in the area of job shop scheduling.

CHAPTER III

EXACT METHODS

Two-machine Job Shop Scheduling Problem with a CDD

A common practice in the scheduling community addresses multi-stage problems by working either with a low number of machines, two in general, and/or a low number of jobs (Conway et al 1967, Sarper 1995, Sung and Min 2001, Gupta et al 2002, Gupta et al 2004, Lauff and Werner 2004c). By using the same approach, this research will address the JSSP E/T with a restricted CDD with two machines.

Formally, there is a quantitative procedure to define whether or not a CDD is restricted or unrestricted for the single machine problem. In this section, such procedures will be extended to the E/T JSSP over a CDD with two machines.

In order to define whether a CDD in a two-machine JSSP is restricted or unrestricted, we will first review the single machine problem. Initially, Kanet (1981) assumed that for the E/T single machine problem, $CDD \geq \sum_{j=1}^{n} p_j$ where p_j is the processing time for job j so that the problem can be solved optimally by using his algorithm SCHED. Later Bagchi et al (1986) showed that Kanet's algorithm was able to reach optimal solutions under the weaker assumption that the CDD $\geq \Delta$, where:

$$\Delta = \begin{cases} p_1 + p_3 + \cdots + p_n & \text{if } n \text{ is odd} \\ p_2 + p_4 + \cdots + p_n & \text{if } n \text{ is even} \end{cases} \qquad \textbf{Equation 4}$$

$$p_1 \leq p_3 \leq \cdots \leq p_n \qquad \textbf{Equation 5}$$

For a CDD in a two-machine JSSP to be unrestricted, two conditions must hold. First, the remaining time to process the final operations on each machine must be enough to apply the SCHED algorithm (Kanet 1981) as if each machine were an unrestricted single machine problem. Second, the completion time of each job's first operation on its corresponding machine has to be less than or equal to the starting time of its subsequent last operation. This starting time is given by the SCHED algorithm. The second condition is equivalent to finding a schedule for each single machine problem where no jobs are tardy. By following the reasoning used by Kanet (1981) and Bagchi et al (1986) the restrictedness of the CDD for the JSSP E/T with two machines can be extended. Generally, in order to minimize the deviation over the CDD for all the jobs, the first operations of each job should have priority on each machine in order to allow subsequent operations to be processed. The set of first operations and the set containing the last operations to be scheduled on each machine before the CDD compete for the time available within the interval from $t = 0$ to $t =$ CDD. If the CDD is "loose enough," say if the CDD $\geq$ PT, where:

$$PT = \sum_{i=1}^{2} \sum_{j=1}^{n} p_{ij} \quad \text{,} \qquad \textbf{Equation 6}$$

(p_{ij} is the processing time for the operation of job j to be performed on machine i with $i = 1, 2$), then, the SCHED algorithm can be applied to the two-machine problem and an optimal solution can be obtained. The closer the CDD to $t = 0$, the tighter (i.e. more restricted) the schedule is. This fact can be used to define whether the CDD is restricted or not. Let

M_1 = Set with jobs to be finished on Machine 1, and

M_2 = Set with jobs to be finished on Machine 2.

$|M_1| = n_1$ = Number of jobs to be finished on Machine 1.

$|M_2| = n_2$ = Number of jobs to be finished on Machine 2.

Also, let

$$\Delta_1 = \begin{cases} p_{11} + p_{13} + \cdots + p_{1n_1} & \text{if } n_1 \text{ is odd} \\ p_{12} + p_{14} + \cdots + p_{1n_1} & \text{if } n_1 \text{ is even} \end{cases} \qquad \textbf{Equation 7}$$

$$\Delta_2 = \begin{cases} p_{21} + p_{23} + \cdots + p_{2n_2} & \text{if } n_2 \text{ is odd} \\ p_{22} + p_{24} + \cdots + p_{2n_2} & \text{if } n_2 \text{ is even} \end{cases} \qquad \textbf{Equation 8}$$

where:

$$p_{11} \le p_{12} \le \cdots \le p_{1n_1} \qquad \textbf{Equation 9}$$

$$p_{21} \le p_{22} \le \cdots \le p_{2n_2} \qquad \textbf{Equation 10}$$

Finally, let

$$F_1 = \sum_{j \in M_1^C} p_{1j} \qquad \textbf{Equation 11}$$

$$F_2 = \sum_{j \in M_2^C} p_{2j} \qquad \textbf{Equation 12}$$

where:

M_1^C is the complement of M_1 and M_2^C is the complement of M_2.

Definition 1

A CDD is unrestricted if CDD $\ge$ *Max* $\{F_1 + \Delta_1, F_2 + \Delta_2\}$ *and the number of tardy jobs in sets* M_1^C *and* M_2^C *are equal to zero.*

Discussion. If Max $\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_1 + \Delta_1$, let $B_1 = \{i = 1, 1 \le j \le n, j \in M_1 \mid C_{1j} \le CDD\}$, where C_{1j} is the completion time of the jobs with their last operation on Machine 1 and to be completed on or before CDD. Similarly, let $A_1 = \{i = 1, 1 \le j \le n, j \in M_1 \mid C_{1j} > CDD\}$, where C_{1j} is the completion time of the jobs with their last operation on Machine 1 and to be completed after CDD. Hence, Δ_1 is the summation of processing times of the jobs in B_1 and by following the reasoning in Kanet (1981), the optimal schedule for the Machine 1 can be obtained by applying his SCHED algorithm to the jobs in sets A_1 and B_1. The starting times of the jobs in sets A_1 and B_1 provide the due dates for their first operations to be processed on Machine 2 (i.e. jobs in set M_2^C). The jobs whose first operation must be performed on Machine 1, jobs in M_1^C, are processed before the jobs in B_1 without interfering with the optimal schedule since CDD $\ge F_1 + \Delta_1$.

If an earliest due date (EDD) sequence yields either zero or one tardy job, then it minimizes the number of tardy jobs (Baker 1974). Therefore, it is possible to find out if the number of tardy jobs in M_1^C is equal to zero by applying the EDD rule to jobs in M_1^C.

Let $B_2 = \{i = 2, 1 \le j \le n, j \in M_2 \mid C_{2j} \le CDD\}$, where C_{2j} is the completion time of the jobs with their last operation on Machine 2 to be completed on or before the CDD and Δ_2 is the summation of processing times of the jobs in B_2. Since $F_2 + \Delta_2 \le F_1 + \Delta_1$, jobs to be finished on Machine 2 can be optimally scheduled by applying Kanet's SCHED algorithm to the jobs in sets A_2 and B_2, where A_2 is defined similar to A_1. In the same way, the starting times of the jobs in sets A_2 and B_2 provide the due dates for their first operations to be processed on Machine 1 (i.e. jobs in set M_1^C). The jobs whose first operation must be performed on Machine 2, jobs in M_2^C, are performed before the jobs in

B_2 without interfering with the optimal schedule. Also, it is possible to find whether the number of tardy jobs in M_2^C is equal to zero by applying the EDD rule to jobs in M_2^C.

The same reasoning can be applied if Max $\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_2 + \Delta_2$. Table 3 shows the processing times and the operation-machine assignment for a five-job example. Figure 5 illustrates the optimal solution for this example when the CDD is unrestricted and equal to 17. In this case Max $\{F_1 + \Delta_1, F_2 + \Delta_2\}$ = Max $\{8 + 9, 9 + 4\} = F_1 + \Delta_1 =$ 17. Note that if the CDD > 17, the problem is still unrestricted.

Table 3 Job Shop Scheduling Example.

Job	Route (Machine Number)		Processing Time	
	Oper. 1	Oper. 2	Oper. 1	Oper. 2
1	2	1	2	4
2	1	2	3	2
3	2	1	3	4
4	1	2	5	4
5	2	1	4	5

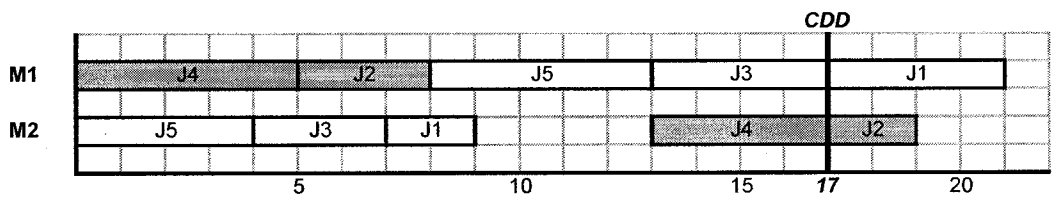


Figure 5 An Example of an Optimal Schedule for the Unrestricted Version.

Definition 2

A CDD is semi-restricted if CDD ≥ Min {$F_1 + \Delta_1$, $F_2 + \Delta_2$} and CDD < Max {$F_1 + \Delta_1$, $F_2 + \Delta_2$} and the number of tardy jobs in the sets M_1^C and M_2^C are equal to zero.

Discussion. If Max {$F_1 + \Delta_1$, $F_2 + \Delta_2$} = $F_2 + \Delta_2$, then Min {$F_1 + \Delta_1$, $F_2 + \Delta_2$} = $F_1 + \Delta_1$. B_1 and B_2 are the same as in Definition 1. Hence, Δ_1 and Δ_2 are the summation of processing times of the jobs in B_1 and B_2 respectively. Given CDD ≥ $F_1 + \Delta_1$, in an optimal schedule, jobs in M_1^C are performed before the jobs in B_1 without interference with the jobs in B_1. The optimal schedule for Machine 1 can be obtained by applying Kanet's SCHED algorithm to the jobs in B_1 and A_1. The starting times of the jobs in sets A_1 and B_1 provide the due dates for their first operations to be processed on Machine 2 (i.e. jobs in the set M_2^C), which are processed before the jobs in B_2. But given that CDD < $F_2 + \Delta_2$, jobs in M_2 cannot be optimally scheduled by using the SCHED algorithm; instead, this problem needs to be treated as a single machine problem with a restricted CDD, which can be optimally solved by using the dynamic programming (DP) procedures proposed by Hall et al (1991). The starting times of the jobs in M_2 given by the optimal solution provide the due dates for their first operations to be processed on Machine 1 (i.e. jobs in the set M_1^C). The jobs whose first operation must be performed on Machine 1, jobs in M_1^C, are processed before the jobs in B_1 without interfering with the optimal schedule since CDD ≥ $F_1 + \Delta_1$. Similar to the unrestricted case, jobs in M_1^C and M_2^C need to be sequenced by using the EDD rule to check if the number of tardy jobs is equal to zero. If both sequences yield zero tardy jobs, then the problem is semi-restricted. Following the same numeric example, Figure 6 describes the case when the CDD is equal

to *15*, which is greater than Min $\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_2 + \Delta_2 = 13$ and is less than Max $\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_1 + \Delta_1 = 17$.

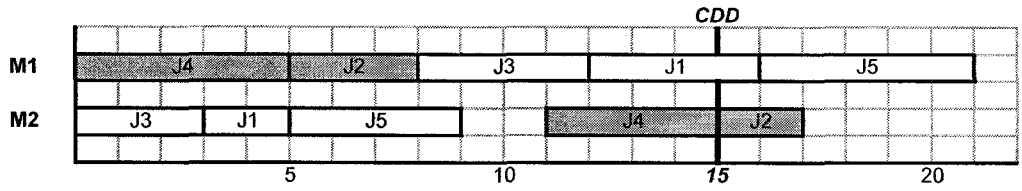


Figure 6 An Example of Optimal Schedule for the Semi-restricted Version.

In this sense, a CDD is semi-restricted when the optimal schedule of either one of the machines can be obtained by using Kanet's SCHED algorithm. The optimal schedule for the other machine has different features, so the SCHED algorithm will not be able to find it. Instead, the DP procedures developed by Hall et al (1991) needs to be used to find the optimal schedule. This procedure is extended to the problem studied here in the next section.

Definition 3

A CDD is restricted when neither the conditions in Definition 1 nor in Definition 2 hold.

Discussion. If CDD < Min $\{F_1 + \Delta_1, F_2 + \Delta_2\}$, let Min $\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_1 + \Delta_1$, and $B_1 = \{i = 1, 1 \le j \le n, j \in M_1 \mid C_{1j} \le CDD\}$, where C_{1j} is the completion time of the jobs with their last operation on Machine 1. Hence, Δ_1 is the summation of processing times of the jobs in B_1. Since CDD < $F_1 + \Delta_1$, there is no way to optimally schedule jobs in M_1 without modifying the starting times of the jobs in M_1^C (the first

operations of jobs in M_2). Hence, a trade off between jobs in M_1^C and jobs in B_1 must be made. The jobs in M_1^C (those with their first operation to be performed on Machine 1) interfere with the optimal schedule on this machine. On the other hand, if at least one of the jobs in M_1^C is delayed, this delay interferes with the optimal schedule on Machine 2. The same reasoning can be applied if Min $\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_2 + \Delta_2$. Following the same numeric example, Figure 7 illustrates the case when the CDD is equal to 8, which is less than Min $\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_2 + \Delta_2 = 13$.

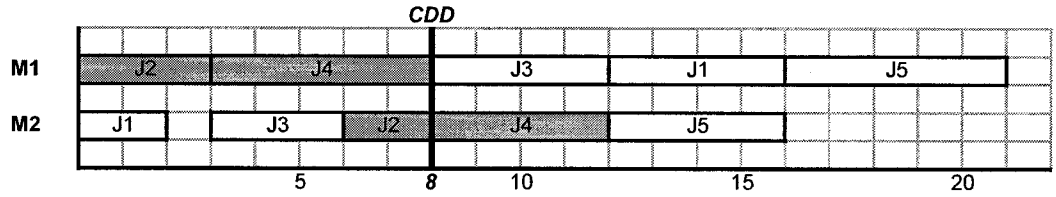


Figure 7 Example of a Final Schedule for the Restricted Version.

Additionally, if there is at least one tardy job in the cases given in Definition 1 or 2, then the CDD is also considered restricted since precedence constraints for at least one job (i.e. the tardy job) do not hold.

Next, optimality conditions for the unrestricted and semi-restricted case are presented. Also, two properties of the optimal solution for the restricted case are proven and used to construct approximate solutions.

Optimality Conditions

Optimal solutions for the two-machine E/T JSSP with restricted CDD appear to be difficult to characterize. In this research, optimal solutions will be defined when the CDD is unrestricted and semi-restricted. Approximate solutions, obtained by a heuristic procedure, will be defined when the CDD is restricted.

Unrestricted CDD

If the CDD is unrestricted as described in Definition 1, the optimal solution can be found by using Kanet's SCHED procedure on each machine. The properties of the optimal schedule as defined in Kanet (1981) will be extended for our use.

Property 1. There is no idle time between jobs in sets M_1 and M_2.

Property 2. The jobs in both B_1 and B_2 are sequenced by longest processing time first (LPT).

Property 3. The last jobs in B_1 and B_2 are completed at time t = CDD.

Property 4. Let A_1 and A_2 represent an ordered set of jobs pertaining respectively to M_1 and M_2 to be scheduled without inserted idle time such that the first job in both A_1 and A_2 starts at time t = CDD. In an optimal schedule, jobs in both A_1 and A_2 are sequenced by the shortest processing time (SPT) first.

Property 5. If n_1 is even, then $|B_1| = |A_1|$. If n_1 is odd, then $|B_1| = |A_1| + 1$. If n_2 is even then $|B_2| = |A_2|$. If n_2 is odd then $|B_2| = |A_2| + 1$.

Property 6. There is a one-to-one mapping of the jobs in both A_1 and A_2 onto the jobs in B_1 and B_2 such that $k_1 \in A_1$ and $k_2 \in A_2$ and $j_1 \in B_1$ and $j_2 \in B_2 \Rightarrow p_{1k_1} \leq p_{1j_1}$ *and* $p_{2k_2} \leq p_{2j_2}$.

The proofs of these properties and the proof that SCHED yields optimal solutions can be easily extended from Kanet (1981) and Definition 1. Figure 5 shows the optimal schedule for the same numerical example when the CDD is unrestricted.

Semi-restricted CDD

If the CDD is semi-restricted as described in Definition 2, the optimal solution can be found by using Kanet's SCHED procedure on the machine with Min $\{F_1 + \Delta_1, F_2 + \Delta_2\}$. The optimal solution in this machine preserves the properties given for the unrestricted case. For the other machine, some properties must be defined in order to characterize the optimal solution. The properties of the optimal schedule as defined by Hall et al (1991) will be extended for our use.

Let t_1^* or t_2^* denote the starting times in an optimal schedule of the first job processed on either M_1 or M_2 respectively corresponding to the machine where Max $\{F_1 + \Delta_1, F_2 + \Delta_2\}$ holds.

Also, define $E_{1(2)} = \{i = 1(2), 1 \leq j \leq n \mid C_{1(2)j} < CDD\}$, $E_{1(2)}' = \{i = 1(2), 1 \leq j \leq n \mid C_{1(2)j} \leq CDD\}$, $T_{1(2)} = \{i = 1(2), 1 \leq j \leq n \mid C_{1(2)j} - p_{1(2)j} \geq CDD\}$, and $T_{1(2)}' = \{i = 1(2), 1 \leq j \leq n \mid C_{1(2)j} > CDD\}$

Property 1. There exists at least one of the following:

An optimal schedule with either $t_1^* = F_1$ or $t_2^* = F_2$.

An optimal schedule with $C_{1(2)a}$ = CDD, where *a* is a job with its last operation on Machine 1*(2)* starting before CDD and completing at CDD or later.

Property 2. In an optimal schedule, the jobs in $E_{1(2)}$' are in LPT order, and the jobs in $T_{1(2)}$ are in SPT order.

Property 3. Each optimal schedule is weakly V-shaped. A weakly V-shaped schedule means a job does not necessarily end at the CDD.

Definition 4

On Machine 1(2) a schedule is early V-Shaped (EVS) if $p_{1(2)a} \leq p_{1(2)tmin}$.

Definition 5

On Machine 1(2) a schedule is tardy V-Shaped (TVS) if $p_{1(2)a} \leq p_{1(2)emin}$.

Where emin = Min $\{ p_{1(2)j} \in E_{1(2)}' \}$, *and tmin = Min* $\{ p_{1(2)j} \in T_{1(2)} \}$.

Property 4. If $t_1^* = F_1$, then $|E_1| \geq |T_1| - 1$ or If $t_2^* = F_2$, then $|E_2| \geq |T_2| - 1$

Property 5. $|E_{1(2)}| \leq |T_{1(2)}| + 1$

Property 6. If $C_{1(2)a}$ = CDD, then $\sum_{j \in E_{1(2)}'} p_{1(2)j} \leq \sum_{j \in T_{1(2)}} p_{1(2)j} + 2p_{1(2)s}$, where job s is the first job scheduled on Machine 1(2).

The proofs of these properties can be easily extended from Hall et al (1991) and Definition 2.

Based on these properties, the DP procedure defined by Hall et al (1991) can be extended to find the optimal solution examining all EVS and TVS schedules if $t_{1(2)}^* = F_{1(2)}$ by using ***EVS*** and ***TVS*** procedures described by Hall et al (1991). If $t_{1(2)}^* > F_{1(2)}$, then it is possible to assume that $C_{1(2)a}$ = CDD and the ***Nosplit*** procedure described by Hall and Posner (1991) can be extended to find an optimal schedule in which a job completes at CDD. By jointly using these procedures, an optimal solution can be found for the two-machine JSSP with a semi-restricted CDD. A more detailed presentation of these procedures will be done in next section. Figure 6 shows the optimal schedule for the same previous numerical example when the CDD is semi-restricted.

Restricted CDD

To characterize the optimal solution when the CDD is restricted can be difficult. Hence, two properties are defined in order to develop a heuristic algorithm to obtain approximate solutions for the two-machine JSSP.

Define:

I_1 = the set of jobs to be finished on Machine 1 and to be scheduled around the restricted CDD.

I_2 = the set of jobs to be finished on Machine 2 and to be scheduled around the restricted CDD.

Clearly, $I_1 \subseteq M_1$ and $I_2 \subseteq M_2$.

Property 1. Jobs in I_1 and I_2 are scheduled without idle time.

Proof. By contradiction and similar to the approach by Baker (1997), assume that an optimal schedule S exists with an idle interval of length t between consecutive jobs a and b, with b following a; $a, b \in I_1$ (I_2), and the predecessors of a and b already scheduled at Machine 2*(1)*. If job a is early ($C_{1(2)a} <$ CDD), then the total penalty cost can be reduced by shifting job a (and any jobs that precede it) later by an amount Δt, where $\Delta t \leq$ Min (t, CDD - $C_{1(2)a}$) without affecting the feasibility of S. Denoting the values after the shift with primes, it follows that $T_{1(2)k}' = T_{1(2)k}$ and $E_{1(2)k}' \leq E_{1(2)k}$ strictly for at least one job. Similarly, if job b is tardy ($C_{1(2)b} >$ CDD), then the total penalty cost can be reduced by shifting job b (and any jobs that follows it) earlier by an amount Δt, where $\Delta t \leq$ Min (t, $C_{1(2)b}$ - CDD) without affecting the feasibility of S. Hence, it follows that $E_{1(2)k}' = E_{1(2)k}$ and $T_{1(2)k}' \leq T_{1(2)k}$ strictly for at least one job. Since any schedule must have either job a early or job b tardy, then schedule S can be improved, and therefore, it cannot be optimal.

Property 2. The optimal schedule for the jobs in I_1 and I_2 is weakly V-shaped, where a schedule S is weakly V-shaped if all jobs completed before the CDD are in decreasing order of their last operation's processing times (LPT), and all jobs that begin their processing after the CDD are in increasing order of their last operation's processing times (SPT).

Proof. By contradiction and similar to the approach in Baker (1997), assume S denotes an optimal schedule in which some adjacent pair of early jobs in $I_{1(2)}$ is not in

LPT order. Then a pair-wise interchange of these two jobs will reduce the total earliness penalty and leave the tardiness penalty unchanged on Machine 1*(2)* without affecting the feasibility of *S*. Similarly, if *S* is an optimal schedule containing an adjacent pair of jobs that starts late in $I_{1(2)}$ and that violates the SPT order, then an adjacent pair-wise interchange will reduce the total tardiness penalty and leave the total earliness penalty unchanged on Machine 1*(2)*. In either case, *S* cannot be an optimal schedule.

Once the jobs to be included in I_1 and I_2 have been defined, there are still two questions to be answered. First, in an optimal schedule, is there some job that must complete exactly at t = CDD? As shown by Hall et al (1991), in the single machine scheduling problem with a restricted CDD, it is not necessary that some job completes exactly at t = CDD. Second, which jobs are to be early, and which ones are to be tardy? Figure 7 shows an approximate schedule of the numerical example when the CDD is restricted.

Dynamic Programming Algorithm for the two-machine JSSP

The algorithm JSSPET presented here uses a DP algorithm to find optimal solutions for the two-machine E/T JSSP when the CDD is semi-restricted. This algorithm partitions the solution space into schedules with either $t_1^* = F_1$ or $t_2^* = F_2$, and those with either $t_1^* > F_1$ or $t_2^* > F_2$. In the first case, jobs either in M_1 or M_2 are scheduled in the interval $[F_1, F_1 + P_1]$ or $[F_2, F_2 + P_2]$, where $P_1 = \sum_{j \in M_1} p_{1j}$ and $P_2 = \sum_{j \in M_2} p_{2j}$. Based on property 3, any optimal schedule is either EVS or TVS, and so ***EVS*** (***TVS***) procedure discussed next will find optimal EVS (TVS) schedules which completes at either $F_1 + P_1$ or $F_2 + P_2$. In the second case, suppose that either $t_1^* > F_1$ or $t_2^* > F_2$ and based on property 1, it is possible to assume that either C_{1a} = CDD or C_{2a} = CDD so that *Nosplit* procedure discussed later will find an optimal schedule in which a job completes at CDD. Finally, the lower cost offered by the three procedures is an optimal schedule. All three procedures make use of properties 2 and 3 (V-shaped structure) and were extended from Hall et al (1991) who addressed the single machine version.

Procedures EVS and TVS consider jobs in non-increasing order. From property 3, job n_1 (to be finished on Machine 1) either starts at F_1 or ends at $F_1 + P_1$, and job n_2 (to be finished on Machine 2) either starts at F_2 or ends $F_2 + P_2$. The total processing time of previously scheduled jobs which finish before CDD in ***EVS*** and after CDD in ***TVS*** procedures are stored. *Nosplit* procedure considers jobs in non-decreasing order based on their processing times.

Procedure EVS

Let $f_k(a_1)$ = the minimum cost to schedule jobs $n_1, n_1\text{-}1, \ldots, n_1\text{-}k+1$ (similarly $n_2, n_2\text{-}1, \ldots, n_2\text{-}k+1$) provided that the latest job scheduled which finished at or before CDD, finishes at time CDD $- a_1$, $a_1 \geq 0$ (similarly CDD $- a_2$, $a_2 \geq 0$), and the earliest job scheduled which finishes after CDD starts at time $\sum_{j=1}^{n_1-k} p_{1j} - a_1 + CDD$ (similarly $\sum_{j=1}^{n_2-k} p_{2j} - a_2 + CDD$). That is, the latest job finishes at $F_1 + P_1$ (similarly $F_2 + P_2$.)

Recurrence relation:

$$f_{k+1}(a_1) = \begin{cases} Min\left\{a_1 + f_k(a_1 + p_{1n_1-k}), \sum_{j=1}^{n_1-k} p_{1j} - a_1 + f_k(a_1)\right\} \\ \qquad\qquad if \sum_{j=1}^{n_1-k} p_{1j} > a_1 \text{ and } a_1 + p_{1n_1-k} \leq CDD \\ a_1 + f_k(a_1 + p_{1n_1-k}) \\ \qquad\qquad if \sum_{j=1}^{n_1-k} p_{1j} \leq a_1 \text{ and } a_1 + p_{1n_1-k} \leq CDD \\ +\infty \\ \qquad\qquad otherwise \end{cases} \qquad \textbf{Equation 13}$$

Boundary condition:

$$f_0(a_1) = 0 \quad for\ a_1 = CDD \qquad \textbf{Equation 14}$$

$$f_0(a_1) = +\infty \quad for\ a_1 \neq CDD \qquad \textbf{Equation 15}$$

Minimum cost schedule defined by:

$$z(\sigma_{EVS}^{*}) = \underset{0 \leq a_1 \leq CDD}{Min} f_n(a_1) \qquad \textbf{Equation 16}$$

Procedure TVS

Let $g_k(m_1)$ = the minimum cost to schedule jobs n_1, n_1-1, ..., n_1-k+1 (similarly n_2, n_2-1, ..., n_2-k+1) provided that the earliest job scheduled which starts at or after CDD, starts at time CDD + m_1, $m_1 \geq 0$ (similarly CDD – m_2, $m_2 \geq 0$), and the latest job scheduled which starts before CDD finishes at time $CDD + m_1 - \sum_{j=1}^{n_1-k} p_{1j}$ (similarly $CDD + m_2 - \sum_{j=1}^{n_2-k} p_{2j}$). That is, the earliest such a job starts is at F_1 (similarly F_2.)

Recurrence relation:

$$g_{k+1}(m_1) = \begin{cases} Min\left\{\left|\sum_{j=1}^{n_1-k-1} p_{1j} - m_1\right| + g_k(m_1),\ m_1 + p_{1n_1-k} + g_k(m_1 + p_{1n_1-k})\right\} \\ \qquad if \sum_{j=1}^{n_1-k} p_{1j} > m_1 \ and \ m_1 + p_{1n_1-k} \leq P_1 - CDD \\ m_1 + p_{1n_1-k} + g_k(m_1 + p_{1n_1-k}) \\ \qquad if \sum_{j=1}^{n_1-k} p_{1j} \leq m_1 \ and \ m_1 + p_{1n_1-k} \leq P_1 - CDD \\ +\infty \\ \qquad otherwise \end{cases} \quad \textbf{Equation 17}$$

Boundary condition:

$$g_0(m_1) = 0 \quad for\ m_1 = P_1 - CDD \quad \textbf{Equation 18}$$

$$g_0(m_1) = +\infty \quad for\ m_1 \neq P_1 - CDD \quad \textbf{Equation 19}$$

Minimum cost schedule defined by:

$$z(\sigma_{TVS}^{*}) = \underset{0 \leq m_1 \leq P_1 - CDD}{Min} g_n(m_1) \quad \textbf{Equation 20}$$

Procedure Nosplit

Let $h_k(e_1)$ = the minimum cost to schedule jobs *1, 2, ..., k* for either M_1 or M_2 without the CDD splitting any job, given that the total processing time of jobs scheduled early (or on time) is either e_1 or e_2.

Recurrence relation:

$$f_{k+1}(a_1) = \begin{cases} Min\left\{e_1 - p_{1\,k+1} + h_k(e_1 + p_{1\,k+1}),\ \sum_{j=1}^{k+1} p_{1j} - e_1 + h_k(e_1)\right\} \\ \qquad\qquad if\ e_1 \geq p_{1\,k+1} \\ \sum_{j=1}^{k+1} p_{1j} - e_1 + h_k(e_1) \\ \qquad\qquad otherwise \end{cases}$$ **Equation 21**

Boundary condition:

$$h_0(e_1) = 0 \quad if\ e_1 = 0$$ **Equation 22**

$$h_0(e_1) = +\infty \quad if\ e_1 \neq 0$$ **Equation 23**

Minimum cost schedule defined by:

$$z(\sigma_{Nosplit}^{*}) = \underset{0 \leq e_1 \leq CDD}{Min}\ h_n(e_1)$$ **Equation 24**

In all of the three procedures, the first alternative in the recurrence relation represents the cost of scheduling the next job as early as possible, and the second one, similarly, as late as possible. Since in the ***TVS*** procedure an early job may finish after the CDD, there is a need for the absolute value in the first equation on its recurrence relation.

JSSPET Algorithm

In order to solve the two-machine E/T JSSP over a CDD, the three cases of the due date must be considered: unrestricted, semi-restricted and restricted. When the CDD is unrestricted, the algorithm JSSPET uses the SCHED procedure (Kanet 1981) to find the optimal solution for both machines. Furthermore, if the CDD is semi-restricted, the algorithm JSSPET jointly uses the SCHED procedure to find the optimal solution for one of the machines (the one where Min $\{F_1 + \Delta_1, F_2 + \Delta_2\}$ holds), and the *EVS*, *TVS*, and *Nosplit* procedures to find the optimal for the other machine. Finally, when the CDD is restricted, the algorithm uses a heuristic procedure called *Restricted_CDD* to find approximate solutions. This procedure reduces a restricted problem to the semi-restricted version by iteratively removing one job at the time. Once the problem is reduced to its semi-restricted version, the SCHED, *EVS*, *TVS*, and *Nosplit* procedures are applied. Since the jobs removed are going to be tardy anyway, the SPT rule is applied in order to minimize their tardiness. Each time a removed job is scheduled, an improvement procedure tries to look for early slots of time in the current schedule in order to decrease the tardiness cost. The pseudo-code of the algorithm JSSPET is given below.

Algorithm JSSPET

Calculate F_1, F_2, Δ_1, Δ_2
Apply SCHED procedure to M_1
Apply SCHED procedure to M_2
Calculate T_1 = Number of tardy jobs in M_1^C
Calculate T_2 = Number of tardy jobs in M_2^C
If CDD $\geq$ Max $\{F_1 + \Delta_1, F_2 + \Delta_2\}$ and $T_1 = 0$ and $T_2 = 0$ then
    Optimal schedule is given by SCHED procedure on both machines
    Schedule first operations on Machine 1 and Machine 2 by using EDD rule.
    Stop.
Else
    Apply SCHED to machine where Min $\{F_1 + \Delta_1, F_2 + \Delta_2\}$ holds

Apply EVS to the other machine
Apply TVS to the other machine
Apply *Nosplit* to the other machine
Solution for the other machine is Min {EVS, TVS, *Nosplit*}
Calculate T_1 = Number of tardy jobs on M_1^C
Calculate T_2 = Number of tardy jobs on M_2^C
If CDD < Max $\{F_1+\Delta_1, F_2+\Delta_2\}$ and CDD ≥ Min $\{F_1+\Delta_1, F_2+\Delta_2\}$ and $T_1=0$ and $T_2=0$ then
Optimal schedule is given by SCHED and Min {EVS, TVS, *Nosplit*}
Schedule first operations on Machine 1 and Machine 2 by using EDD rule.
Stop.
Else
Apply Restricted_CDD
End If
End If
End Algorithm

Computational Experiments

The JSSPET algorithm was implemented in Basic language (version 6.0) and tested on a 3.00 GHz Pentium IV running Windows XP™.

Sets of problems with 2 machines; 5, 6, 7, 8, 10, 20, 50, 100 and 500 jobs; with 30 problem instances per problem size were generated. The processing times were generated from a discrete uniform distribution U(1, 100), and the jobs routes were obtained from another discrete uniform distribution U(1, *m*) where *m* is the number of machines (two in our case). Similar to most random numbers generators use today, the processing times and jobs routes were generated by using random numbers coming from a linear congruential generator (Law and Kelton 2000).

The CDD for the unrestricted case is given by equation 25:

$$CDD = Max(F_1 + \Delta_1, F_2 + \Delta_2) \qquad \textbf{Equation 25}$$

For the semi-restricted case, the CDD is chosen to be in the middle of the interval between $Min(F_1 + \Delta_1, F_2 + \Delta_2)$ and $Max(F_1 + \Delta_1, F_2 + \Delta_2)$ as given by equation 26:

$$CDD = Min(F_1 + \Delta_1, F_2 + \Delta_2) + 0.5 * [Max(F_1 + \Delta_1, F_2 + \Delta_2) - Min(F_1 + \Delta_1, F_2 + \Delta_2)] \qquad \textbf{Equation 26}$$

Finally, for the restricted case, the CDD is given by equation 27:

$$CDD = \lfloor h * Min(F_1 + \Delta_1, F_2 + \Delta_2) \rfloor \qquad \textbf{Equation 27}$$

where h is the tightness factor that takes four possible values, $h = 0.7, 0.8, 0.9, 0.95$ and $\lfloor x \rfloor$ is the largest integer less than or equal to x.

Considering the four cases of the restricted version, the single case of both the unrestricted and the semi-restricted version, a total of 9 * 30 * 6 = 1620 problem instances were solved.

Results

The results in Table 4 show the average, standard deviation, and maximum computational solution times for each set of instances. The times are in seconds and exclude input and output time. Computational solution times increase approximately linearly with n for the unrestricted case, and in proportion to n^2 for the semi-restricted case. These results confirm that *JSSPET* algorithm finds optimal solutions for both the unrestricted and the semi-restricted cases for large random instances of the problem within an average of at most 20 minutes. Such result is made possible by the new optimality conditions extended from the single machine problem provided in this research, which enable us to prove the optimality of the dynamic programming procedure. Data and optimal solutions for instances of unrestricted and semi-restricted problems up to 500 jobs will be made available at www.schedulingresearch.com.

For restricted problems with 5, 6, 7, and 8 jobs, finding an optimal solution is not guaranteed, but when compared with optimal solutions obtained through a mixed integer linear programming (MILP) formulation, it turned out the JSSPET algorithm found optimal solutions for many instances by applying the *Restricted_CDD* procedure. Table 4 shows the Average Deviations (*AD*) from the optimal solutions and their Standard Deviations (*SD*) for 30 instances per job size. The deviation is calculated as follows:

$$AD = [(JSSPET\ Solution - Optimal\ Solution)/Optimal\ Solution]\ X\ 100\%. \quad \textbf{Equation 28}$$

In Table 4, *AD* ranges from 8% to 26% but in about 30% to 40% of all the instances *AD* is less than 5% from the optimal value. Data and both optimal and heuristic solutions will be available at www.schedulingresearch.com.

For restricted problems with 10 or more jobs, the *JSSPET* algorithm is evaluated based on how far its solutions are from a lower bound (*LB*). The *LB* used in this case is the optimal solution for the same instances but with a semi-restricted CDD. Recall that a problem instance's solution with a restricted CDD will always be larger than the same instance with unrestricted or semi-restricted CDD. Table 5 shows the Average Deviations (*AD*) from the *LB* (the semi-restricted version) and their Standard Deviation (*SD*). *AD* is calculated as follows:

$$AD = [(JSSPET\ Solution - LB\ solution)/\ LB\ solution]\ X\ 100\%. \quad \textbf{Equation 29}$$

As shown in Tables 5 and 6, as the tightness factor *h* decreases, the CDD becomes tighter (i.e. problem is more restricted), and so *AD* increases. Since the behavior of the optimal objective function value for the restricted version of the problem is unknown, using the solution of the semi-restricted version as an LB for the restricted one tends to underestimate the performance of the *JSSPET* algorithm. Therefore, this LB needs to be used carefully, and a better LB can be pursued in future research. Note that in Table 5, for small problems with restricted CDD and when n = 6 jobs, the average deviation is higher than other problems, and this result was due to having 80% of the 30 random instances unbalanced and 20 % balanced. Therefore, the performance of the heuristic was worse than other problem sizes.

Table 4 Computational Times for the Unrestricted and Semi-restricted Cases.

Jobs	Unrestricted			Semi-restricted		
	Average (sec.)	*Deviation (sec.)*	*Maximum (sec.)*	*Average (sec.)*	*Deviation (sec.)*	*Maximum (sec.)*
n = 5	0.004	0.007	0.016	0.6230	1.1485	2.6698
n = 6	0.004	0.007	0.016	0.7674	1.2943	2.8778
n = 7	0.006	0.008	0.016	1.1335	1.5152	3.0914
n = 8	0.006	0.008	0.016	1.2123	1.6205	3.3062
n = 10	0.006	0.008	0.016	1.4929	1.8597	3.7322
n = 20	0.014	0.005	0.016	4.9724	1.9837	5.7373
n = 50	0.029	0.007	0.047	19.9488	5.5026	32.9513
n = 100	0.056	0.008	0.063	67.1799	9.2983	74.6444
n = 500	0.294	0.014	0.359	1299.8607	63.9390	1590.2551

Table 5 AD and SD for the Small problems, Restricted Version.

Jobs	Restricted							
	$h = 0.70$		$h = 0.80$		$h = 0.90$		$h = 0.95$	
	AD	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*
n = 5	11.72%	12.54%	11.02%	20.20%	12.79%	11.49%	19.49%	20.59%
n = 6	15.68%	17.29%	18.19%	19.30%	22.25%	21.84%	26.00%	32.43%
n = 7	13.10%	10.28%	12.88%	8.00%	17.82%	12.53%	21.89%	17.83%
n = 8	14.05%	11.30%	11.09%	12.43%	12.15%	12.75%	18.09%	20.89%

Table 6 AD and SD for the Large problems, Restricted Version.

Jobs	Restricted							
	$h = 0.70$		$h = 0.80$		$h = 0.90$		$h = 0.95$	
	AD	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*
n = 10	95.18%	18.27%	60.03%	12.83%	36.93%	15.13%	28.42%	10.81%
n = 20	77.45%	26.84%	49.82%	19.47%	28.11%	17.64%	20.83%	17.63%
n = 50	55.22%	18.17%	34.18%	18.57%	18.78%	15.89%	12.98%	15.38%
n = 100	44.76%	14.29%	24.26%	10.19%	10.73%	9.93%	6.06%	7.28%
n = 500	34.52%	4.43%	17.07%	3.33%	6.06%	2.05%	2.73%	1.41%

The Balance Ratio (*BR*) can be defined as the ratio between the summations of the processing times of the operations to be performed before the CDD in Machine 1 and in Machine 2 as shown in equation 30:

$$BR = (F_1 + \Delta_1)/(F_2 + \Delta_2) \qquad \textbf{Equation 30}$$

The closer *BR* is to one, the more balanced the problem is. In a balanced problem, the total amount of processing time before the CDD in both machines is similar, allowing for an even utilization of both machines and then decreasing the value of the objective function. For the restricted version of the problem, the more compact the schedule is, the less the tardiness cost is, and hence, the less the objective function value. For balanced problems, the *JSSPET* algorithm is able to find compact schedules with less tardiness costs. On the other hand, the *JSSPET* algorithm underutilizes the machines when the problem is less balanced, increasing the tardiness costs. The relationship between *BR* and *AD* is shown in Figure 8. Either Machine 1 or 2 is underutilized depending on the value of *BR*. If $BR > 1$, then Machine 2 is underutilized, and if $BR < 1$, then Machine 1 is underutilized. Because of this behavior, the relationship takes the form of a quadratic trend line equation with correlation coefficients close to 90% for the sets with 50, 100 and 500 jobs.

Additionally, it can be seen that for 50, 100 and 500 jobs, the more balanced the problem is, i.e. the closer BR is to 1, the smaller *AD* becomes, and as the number of jobs increases, *AD* decreases. Low *AD* values for larger number of jobs can be explained by

the fact that the higher the numbers of jobs, the higher the chances are to find processing times from the uniform distribution, and therefore, more balanced loads on both machines. For these problems, *JSSPET* finds solutions closer to the LB.

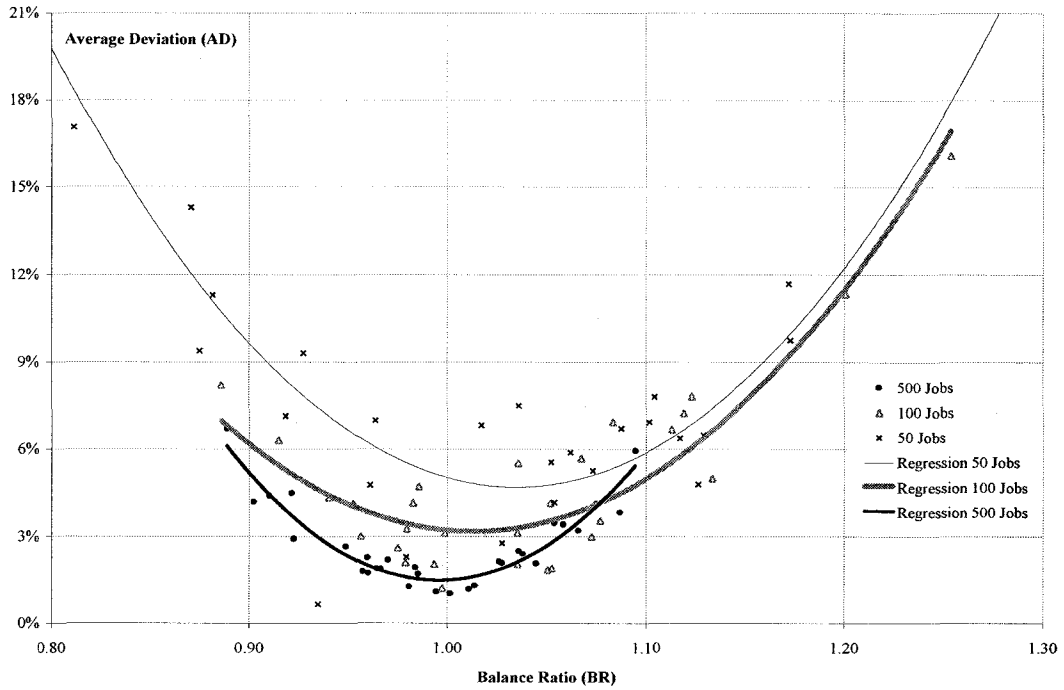


Figure 8 Relationship between AD and BR.

CHAPTER IV

MULTI-AGENT SYSTEMS

Distributed Computing (DC) has recently been used to solve complex scheduling problems that arose in both industry and theory. As stated by Sousa and Ramos (1999), scheduling of manufacturing systems matches a distributed problem from the physical and from the logical point of view. In this sense, DC has already given some answers to the problem of how to efficiently implement communities of interactive systems. A new research area has appeared to cover the problem posed by the integration of Artificial Intelligence (AI) and (DC); this area is the Distributed Artificial Intelligence (DAI).

Davis and Smith (1983) suggested that DAI methodologies lead to two different approaches: Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS). In DPS, there is a set of modules or nodes co-operating to solve a specific problem. The knowledge about the problem and its solution is divided among all nodes of the system. In MAS, however, the distinction between problem solving and co-operation is much clearer. The attention is on the coordination process between intelligent autonomous agents. The negotiation between different agents is one of the most important problems to solve in DAI.

Two main processes can be considered as the most important in a DAI approach: control and communication (Decker 1987). Based on the characteristics of the control process, Crowe and Stahlman (1995) defined three types of controlling tasks: hierarchical, heterarchical and quasi heterarchical as described in Figure 9. They proposed the last one to combine many of the advantages of hierarchy and heterarchy,

with few of their disadvantages. In their approach, four categories of distributed control strategies have been identified: sequencing, bidding, negotiation and cooperation.

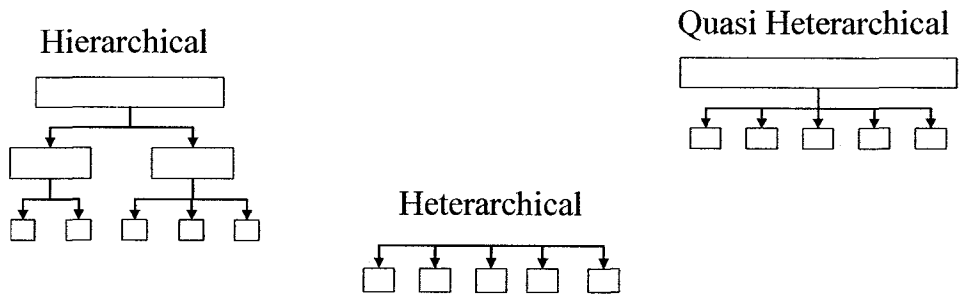
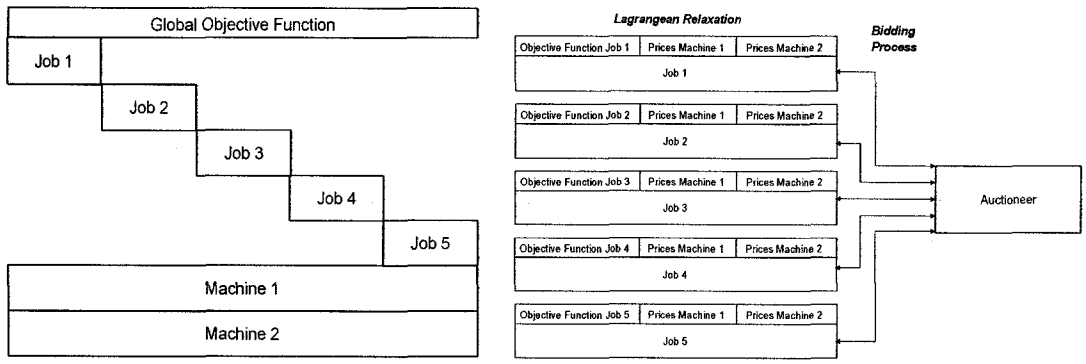


Figure 9 Controlling tasks in DAI (Crowe and Stahlman 1995).

As was shown in Table 2, most of the research carried out concerning JSSP, solved by using a multi agents approach defines jobs and machines as agents, regardless of which control strategy is executed. Pinedo (2002) proposed a general framework to describe what he named Market-based and Agent-based procedures for scheduling. In his framework, both jobs and machines are considered as agents interacting in a market, where *Job Agents* need specific tasks to be performed, and *Machine Agents* have the capacity to carry out those tasks.

The approach proposed in this research is based on a decomposition method, which uses an adapted version of Lagrangean relaxation suitable to handle iterative auctions. The basic idea is to localize and distribute the operational scheduling decisions, leaving the complexity to local decision makers, while maintaining a simple and generic coordination mechanism at a central site. This approach is considered distributed since each local decision maker supports their decisions on a local utility, which is based on both local preferences and global constraints. Specifically, each decision maker has a

local problem to maximize their expected total reward which is subject to local constraints. This is then communicated to the central coordinator as a "bid." The central coordinator (the auctioneer) is a bid processor that makes resource allocation based on an iterative auction process using the bidding information. Figure 10 shows the proposed approach with a job shop scheduling problem considering 5 jobs and 2 machines.



Global Optimization Formulation Decentralized Formulation

Figure 10 Decomposition strategy for a job shop scheduling problem.

An integer linear programming formulation (ILPF) of the job shop scheduling problem is used to schedule jobs by using combinatorial auctions in a distributed control fashion. Initially, the ILPF needs to be relaxed by an easier to solve and separable version. In general terms, relaxing a linear problem means to replace a set of complicated constraints with a penalty term in the objective function involving the amount of the violation of these constraints. Then, the relaxed problem can be solved, and it provides either upper or lower bounds on the optimal solution of the original problem (Fisher 1985). This method is known as Lagrangean relaxation, and it is used to replace the machine constraints in the global formulation of the ILPF (See Figure 10)

with a penalty term in the objective function. This new problem is separable in terms of jobs, and it is easier to solve. Once the original problem has been separated at the job-level, each one of these problems can be optimally solved in an independent way. Based on this solution, each job agent bids for a specific subset of resources in an auction procedure. As shown, the Lagrangean relaxation is used as a way to separate the problem, not only facilitating its solution, but also allowing for a parallel procedure to solve it. The combinational auctions procedure takes advantage of this parallelization to find optimal or quasi-optimal solutions for the global job shop scheduling problem.

This approach can be classified as having a quasi-hierarchical control. There is an auctioneer who controls the pricing process, and jobs agents are bidding for the resources as shown in Figure 10. By using a bidding process, prices of the resources are updated by the auctioneer in an iterative procedure.

Some auctions sell many assets simultaneously. As Rothkopf et al (1998) once pointed out, the assets and their bids are different, depending on which other assets the bidder wins. For instance, in the radio spectrum auctions, a license for the Philadelphia region may be much more valuable to a company if it also has a license for the New York and/or the Washington region (Rothkopf et al, 1998). In this situation, the value of an asset is increased if another group of assets is won. Because of this fact, when simultaneous sales are designed, allowing single bids not only for individual assets but for combinations of assets, the possibility of synergy in values could increase. These kinds of bids are called combinational bids, and the auction process is known as combinational auctions (Rothkopf et al, 1998).

Likewise, in a job shop scheduling problem, job agents may demand a combination of resources (most of the times machines) to process their operations. Hence, the proposed approach needs to deal with a situation in which job agents bid for multiple resources that have interdependent valuations. In a combinational auction, bidders demand a set of indivisible objects with a single bid.

As an example, consider the job shop scheduling problem already presented in Table 3 with 2 machines, 5 jobs, and an unrestricted CDD equal to 17. In this problem, job 1 should bid for 2 continuous time slots from Machine 2 (t= 1 to t= 2), and 4 on Machine 1 (t= 14 to t= 17) in a single bid. For job 1 to be completed on the CDD (t= 17), time slots on Machine 1 and 2 have interdependent values. Also, job 1 is required to win both sets of time slots in order to maintain the technological order and non-preemption constraints since its completion requires processing time on both machines in its given technological order. In a similar way, the remaining jobs have interdependent values for their time slots. Even more, jobs 1, 3, and 5 will compete for time slots on Machine 1 close to the CDD, and Jobs 2 and 4 will compete for similar time slots on Machine 2.

In most of the resource allocation problems, competitive equilibrium prices are known to exist and auction procedures are aimed to reach one of these equilibrium prices in an efficient manner. For assignment problems, Bertsekas (1988) had shown that the prices obtained at the end of the bidding process are the approximate optimal dual ones of the primal problem. However, Wellman et al (2001) have shown that these prices may not exist in a general combinational auction when agents demand a bundle of interdependent and indivisible objects. A set of conditions needs to be met for the

equilibrium prices to exist. Wellman et al (2001) enumerated three general conditions for a resource allocation problem to have equilibrium prices as follows:

- Agents (job-agents in this research) make their own decisions about how to bid based on the prices and their own relative valuations of the goods (time slots). It means they can make effective decisions with local (private) information, without knowing the private information and strategies of other agents.
- Communication is limited to the exchange of bids and prices between agents and the auctioneer.
- In specific cases, the auctioneer can reveal the information necessary to achieve the optimal or come within some tolerance of the optimal.

Methodological approach

In this chapter, the job shop scheduling problem with earliness and tardiness (JSSP E/T) over a common due date (CDD) will be addressed using an auction-based method. In this research, a dynamic or progressive mechanism is proposed (Demange et al 1986) since a number of iterations are carried out before allocating objects to bidders. Objects are the discrete time slots on the machines, and bidders are the job agents. Also, an auctioneer is the coordinating agent or the seller who interactively updates the prices of the resources. Based on the current prices, each job agent tries to find the best combination of time slots on the machines so as to maximize their own utility function. The auctioneer evaluates bids from all the jobs and updates the reservation prices (the maximum price job agents are willing to pay for the time slots) after resolving the conflicts among their requests. This process is repeated in an iterative way until a conflict-free allocation is found. As expected, job agents have inter-dependent values, and different combinations of time slots present different values. Further, note that in a JSSP, precedence and non-preemption constraints restrict the combinations of time slots on which each job can bid.

In the worst case, an auctioneer offering n assets could receive bids on $2^n - 1$ different combinations of assets (this follows from the fact that the total number of distinct subsets on a set of n elements is given by the binomial sum $\sum_{k=0}^{n}\binom{n}{k} = 2^n$, and the empty set is not considered). Bid evaluation could present a computational problem when n is large. Moreover, it has been shown that finding the revenue-maximizing set of non-conflicting bids is a NP-Hard problem by itself (Rothkopf et al 1998).

Problem Formulation:

In order to implement the auction approach, first, the integer linear programming formulation needs to be presented. The following formulation was modified from Pritsker et al (1969) to incorporate earliness and tardiness within the problem. The notation listed below is adopted:

i = job agent (bidder) index, $i = 1,\ldots, n$, where n is number of jobs

j = operation index, $j = 1,\ldots, n_i$, where n_i is the number of operations of job i

t = time slot index, $t = 1,\ldots, T$, where T represents the length of the planning horizon during which all the jobs can be completed

k = machine index, $k = 1,\ldots, m$, where m is number of machines (therefore there are $T*m$ time slots for bidding)

CDD = Common due date

o_{ik} = the operation of job i that requires machine k

m_{ij} = machine required for operation j of job i, $o_{ik} = j$ if $m_{ij} = k$

p_{ij} = processing time for operation j of job i

r_i = release time of job i

$B_{ij;a,b}$ = operation bid, a combination of time slots from time slot a to time slot b for operation j of job i

B_i = job bid, a collection of operations bids (a combination of time slots demanded by job i)

The time slots available from the machines can be defined as a set of pairs (machine, time slot). Therefore, each possible bid B_i from job agent i is a subset of the following object set:

$O = \{(k, t): 1 \leq k \leq m, 1 \leq t \leq T\}$

Also, each operation bid $B_{ij;a,b}$ is a subset of machine m_{ij}'s object set:

$O_{m_{ij}} = \{(m_{ij}, t): 1 \leq t \leq T\}$

Since preemption of operations is not allowed, the operation bid is restricted as follows:

$B_{ij:a,b} = \{(m_{ij}, t): 1 \leq a \leq t \leq b \leq T, b = a + p_{ij} - 1\}$

Thus, job i's overall bid is a limited combination of allowed operation bids:

$$B_i = \bigcup_{j:a_{i,j+1} > b_{i,j}} B_{ij:a_{ij},b_{ij}} \qquad \textbf{Equation 31}$$

Precedence constraints between consecutive operations are defined in the condition of the set definition.

For example, using the JSSP presented in Table 3 and with the same CDD, job 1 needs to send two bids to the auctioneer as follows:

$B_{1,1:1,2} = \{(m_{11} = 2, t): 1 \leq 1 \leq t \leq 2 \leq T, b = 1 + (p_{11} = 2) - 1\}$

$B_{1,2:14,17} = \{(m_{12} = 1, t): 1 \leq 14 \leq t \leq 17 \leq T, b = 14 + (p_{12} = 4) - 1\}$

Note that the bid itself guarantees both the precedence constraint and the non-preemption constraint for job 1.

Recall that in this research, a job incurs in a tardiness cost if its completion time is after the CDD and in an earliness cost when its completion time is before the CDD. Therefore, job i's utility function can be defined as follows:

$$U_i(B_i) = -(\alpha_i E_i(B_i) + \beta_i T_i(B_i)) - P_i(B_i)$$ **Equation 32**

where $P_i(B_i)$ is the total payment if the demanded time slots in bid B_i were awarded to job i. The first two terms account for the total unweighted earliness and tardiness cost attributed to job i by demanding B_i (in our case, α_i and β_i equal to 1 for all jobs). Note that each job must trade off possible savings on CDD performance with payments due to resource usage. The best bid for job i is one that maximizes the utility function defined above, which in equation 32 could be better explained as a cost minimization problem as defined in Equation 33.

$$C_i(B_i) = (\alpha_i E_i(B_i) + \beta_i T_i(B_i)) + P_i(B_i)$$ **Equation 33**

Based on the notation given, the decision variable X_{ijt} is defined as:

$X_{ijt} = 1$ if operation j of job i completes in time period t; 0 otherwise. $Y_{ijm} = 1$ if operation j of job i is processed on machine m. ET_i is either the earliness or tardiness cost (a job cannot be early and tardy at the same time). The objective function and constraints are:

$$Min \quad \sum_{i=1}^{n} ET_i$$ **Equation 34**

s.t.

$$ET_i \geq \sum_{t=1}^{T} tX_{in_i t} + p_{in_i} - CDD \quad \forall i$$ **Equation 35**

$$ET_i \geq CDD - (\sum_{t=1}^{T} tX_{in_i t} + p_{in_i}) \quad \forall i$$ **Equation 36**

$$\sum_{t=1}^{T} X_{ijt} = 1 \quad \forall i, j$$ **Equation 37**

$$_{ik}$$

$$\sum_{t=1}^{T} tX_{ijt} + p_{ij'} \le \sum_{t=1}^{T} tX_{ij't} \quad \forall i, j, j' = j+1 \qquad \textbf{Equation 38}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n_i} X_{ijt}Y_{ijm} + \sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{t'=t+1}^{Min\{T,\,t+p_{ij}-1\}} X_{ijk'}Y_{ijm} \le 1 \quad \forall m.k \qquad \textbf{Equation 39}$$

$$\sum_{t=1}^{T} tX_{i1t} \ge p_{i1} + r_i \quad \forall i \qquad \textbf{Equation 40}$$

$$X_{ijt} \in \{0,1\} \quad \forall i, j, t. \qquad \textbf{Equation 41}$$

Equation (33), the objective function, minimizes the penalty of either earliness or tardiness cost based on the completion time of the last operation n_i for job i in a period t. Constraint sets (34) and (35) imply the minimization of the absolute deviation from the CDD. Constraint set (36) implies that each operation for a job can be completed in only one time period. These are individual knapsack constraints, one for each operation of each job. Constraint set (37) is the precedence constraints for each job, and these constraints ensure that the completion times of the two consecutive operations are separated by the processing time of the later operation. Constraint set (38) ensures that, for each machine, in each time period, the capacity of the machine is not violated. Constraint set (39) ensures that the first operation cannot be completed before the job has been in the shop for at least the processing time of the first operation. Constraint set (40) defines the binary decision variables.

Algorithm

Bid construction. A useful computational idea from the 1970s is the observation that many hard combinatorial optimization problems can be viewed as easy problems complicated by a relatively small set of constraints. Dualizing these constraints (i.e. adding them as penalty terms into the objective function) produces a Lagrangean problem that is easy to solve and whose optimal value is a lower bound, for minimization

problems, on the optimal value of the original problem (Fisher 1981). Here, this idea is used to relax the machine capacity constraints in order to obtain pricing information included in the penalty terms, as well as a decomposable set of jobs problems. This decomposition allows each job to act as an intelligent agent that attempts to find its way through a shop comprised of machines. All job agents in the shop have the ability to communicate directly with a central coordinator, i.e., the auctioneer managing the set of machines. The goal of each job agent is to maximize its own utility. The utility for the central coordinator is earned from the job agents' demand on the machines. If there were no conflicts in resource requirements for each job agent, then each one of them could compute completion times on each machine, and putting together these individually created job schedules could create the entire schedule. However, in a heavily loaded system, there will always be resource competition; hence, it is unlikely to create a feasible schedule in this manner.

A mechanism that can align the actions of these job agents in a global direction is required. By relaxing the machine capacity constraints, a Lagrangean Dual (LR) problem is created and the price of the time slot t on machine k is recorded in a vector λ_{kt}. Once, λ_{kt}, the prices vector, is given, the objective of LR is separable for each job. Furthermore, (4), (5), (6), (7), and (8) are already separable in terms of jobs. Then, LR can be decomposed into sub-problems for each job containing a piece of objective of LR (See Figure 1) and a subset of constraints from the global sets (4), (5), (6), (7), and (8). Given λ_{kt}, the sub-problems can be solved independently since they do not interact. The best choice of λ_{kt} would be the optimal solution to the dual problem.

Bid calculation. Assume a price vector λ_{kt} has been assigned to each machine k for each time slot t. Each job needs to determine the minimum cost, as well as the time to be completed and to leave the shop, subject to precedence, non-preemption, and arrival time constraints. Once the solution of the job sub-problem is found, it is used to compute bids. Time slots on a particular machine that the job requires are the objects desired in the auction; a bid is nothing but the objects desired and the amount each job agent is willing to pay for the objects. The objects requested by the job agent are intervals comprising start time and completion time on each machine that it requires to be completed. The amount the job is willing to pay is the minimum cost computed as the objective of the job sub-problem. Thus, the bid can easily be constructed once the job sub-problem is solved.

Price calculation. The purpose of price adjustment is to ensure that the resource prices λ_{kt} are at an optimal level, beyond which a job agent finds it too expensive to use, while at the same time maximizing the revenue the auctioneer earns. The revenue earned by the auctioneer is a function of the amount the jobs are willing to pay for the objects' bid. If all the resource prices λ_{kt} are set to zero for each time slot on each machine, it implies that the cost of increasing capacity on each one of them is zero, and the schedule generated will have a lot of overlaps. On the other hand, if the value of λ_{kt} is increased sufficiently, then a schedule will be generated where the job completion times are spread out due to different cost structures for each job agent. The adjustment of price for time slots also has an economic interpretation. If there are multiple bidders that desire the same object, then it is to sellers' advantage to increase the price until it is equal

to the highest valuation of the object among all bidders. However, the auctioneer does not have knowledge of the highest valuation because of asymmetry of information imposed by the distributed architecture. This is where auctions are helpful to maximize the revenue earned from objects with unknown valuation. At each round of bidding, checking the overlap for each time slot on each machine determines the direction in which the prices need to be adjusted. If more than one job demands the same object, the price of the slot can be increased. On the other hand, if the price is increased too much and all the bidders find it too expensive to bid, then the price needs to be reduced. Using the direction of surplus or deficit of demand to adjust prices can result in reducing resource contention. A step size can be used for increasing or decreasing the prices in successive iterations in the direction of surplus or deficit. The goal of price updates is to reduce resource conflicts when the same time slot is demanded by more than one job agent. As it was explained, a bid defines a set of objects (time slots) demanded for each job agent, so one way to update prices is to adjust them according to excess job demand, i.e. number of job agents that bid for a certain time slot minus the total capacity of the machine. Generally, the auctioneer raises the prices in proportion to excess demand as follows:

$$D_{kt} = \sum_{i=1}^{n} \delta^{*}_{ikt} - 1 \qquad \textbf{Equation 42}$$

where δ_{ikt}^{*} is 1 if job i demands time slot (k, t) in its optimal bid, 0 otherwise. Since excess demand can be negative, it is possible to reduce prices for time slots which

do not have enough demand. Given that no object can be sold with a negative price, only nonnegative prices are considered and the strategy to adjust prices can be defined as follows:

$$\lambda_{kt}^{r+1} = Max\{0, \lambda_{kt}^{r} + f(D_{kt}^{r})\} \qquad \textbf{Equation 43}$$

where r is the iteration number, and f is the price adjustment function increasing in current excess of demand D_{kt}^{r}. Based on the form of f, different auction protocols that govern the progress can be defined.

When the function f is defined as a constant multiplier times the current excess demand,

$$f(D_{kt}^{r}) = sD_{kt}^{r} \qquad \textbf{Equation 44}$$

where s is called the price adjustment factor or step parameter; the auction protocol is named the *standard Walrasian tâtonnement*. This protocol, under a pure exchange economy with continuous demand is known to converge to an optimal (equilibrium) allocation. When the auctioneer makes aggressive price updates in early iterations to quickly assess the overall demand status among jobs agents, the protocol is named *adaptive tâtonnement*. The protocol follows smaller adjustments (low s values) in later iterations to fine tune the quality of allocation. On the other hand, when there is no price discrimination, the protocol is named *regular tâtonnement*, otherwise (with price

discrimination) named *augmented tâtonnement*. Based on the definition of the price adjustment function, there are two types of auction protocols, *standard* and *adaptive*. Also, based on the payment function, there are two alternative utility functions, *regular* and *augmented*.

In order to use an *augmented tâtonnement*, some properties of the JSSP E/T CDD's optimal solution need to be stated. To characterize the optimal solution when the CDD is restricted, two properties are defined. Let's define I_m as the set of jobs to be finished on machine m, to be scheduled around the restricted CDD. Also, let M_m be the set of jobs to be finished on machine m. Clearly, $I_m \subseteq M_m$.

Property 1. Jobs in I_m are scheduled without idle time.

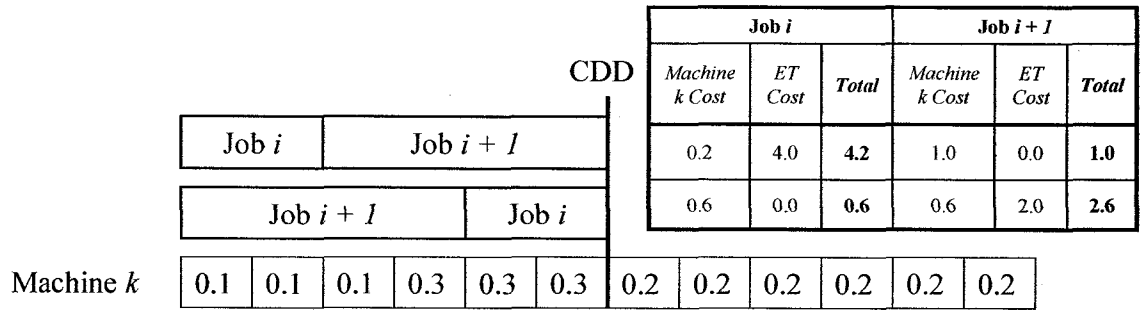
Proof. By contradiction and similar to the approach used by Baker (1997), and similar to the two-machine proof presented earlier in this dissertation, assume that there exists an optimal schedule S with an idle interval of length t between consecutive jobs a and b to be finished on machine m, with b following a; $a, b \subseteq I_m$, and all the predecessors of a and b already scheduled at the first $m - 1$ machines. If job a is early, its completion time on machine m is less than the CDD, i.e. $C_{ma} <$ CDD, then the total penalty cost can be reduced by shifting job a (and any jobs that precedes it) later by an amount Δt, where $\Delta t \leq$ Min (t, CDD - C_{ma}) without affecting the feasibility of S. Denoting the values after the shift with primes, it follows that $T_{mk'} = T_{mk}$ and $E_{mk'} \leq E_{mk}$ strictly for at least one job to be finished on machine m. Similarly, If job b is tardy ($C_{mb} > CDD$), then the total

penalty cost can be reduced by shifting job b (and any jobs that follows it) earlier by an amount Δt, where $\Delta t \leq$ Min (t, C_{mb} - CDD), without affecting the feasibility of S. Hence, it follows that $E_{mk'} = E_{mk}$ and $T_{mk'} \leq T_{mk}$ strictly for at least one job to be finished on machine m. Since any schedule must have either job a early or job b tardy, then schedule S can be improved, and therefore, it cannot be optimal.

Property 2. The optimal schedule for the jobs in I_m is weakly V-shaped, where a schedule S is weakly V-shaped if all jobs completed before the CDD are in decreasing order of their last operation's processing times (LPT), and all jobs that begin their processing after the CDD are in increasing order of their last operation's processing times (SPT).

Proof: By contradiction and similar to the approach in Baker (1997), and similar to the two-machine proof presented earlier in this dissertation, assume S denotes an optimal schedule in which some adjacent pair of early jobs in I_m is not in LPT order. Then, a pairwise interchange of these two jobs will reduce the total earliness penalty and leave the tardiness penalty unchanged on machine m without affecting the feasibility of S. Similarly, if S is an optimal schedule containing an adjacent pair of jobs that starts late in I_m and that violates the SPT order, then an adjacent pairwise interchange will reduce the total tardiness penalty and leave the total earliness penalty unchanged on machine m. In either case, S cannot be an optimal schedule.

Based on these two properties, a price discrimination process can be defined. It is easy to note that there will be more bids on slots closer to the CDD than on the others. Therefore, in order to prevent some jobs agents from competing for those time slots, a higher price could help to better allocate these time slots. Two facts should be considered. First, there is a zone on each machine k [CCD – w_k, CDD + w_k] where $\lambda_{kt} \neq 0$, and second, there will always be job agents willing to pay for being processed during that time interval. Job agents finishing on each machine k with shorter processing times will find it more profitable to bid higher for time slots on that interval. Figure 11 shows different alternatives when two jobs, to be finished on machine k, are bidding for time slots close to the CDD. Note that the prices are given by the iterative price calculation step in the bidding process. The processing time of Job i's last operation is 2 while for Job $i+1$ is 4. In this case, both of them want to be finished on time; their costs (including the costs associated to their final operations and the earliness costs) are shown in the upper right part of the graph. Clearly, job i will be willing to pay up to 3.6 more (1.8 per time slot) to win the two time slots before the CDD while job $i+1$ is willing to raise its bid 1.6 more (0.4 per time slot). Hence, it is to the auctioneer's advantage to place a higher price for the time slots close to the CDD so that job agents with higher value can get the slots. The auctioneer might be willing to increase the prices placed for time slots on machine k in the zone [CDD – 2, CDD + 2] to make additional profits.



	Job *i*			Job *i* + *1*		
	Machine k Cost	*ET Cost*	***Total***	*Machine k Cost*	*ET Cost*	***Total***
	0.2	4.0	**4.2**	1.0	0.0	**1.0**
	0.6	0.0	**0.6**	0.6	2.0	**2.6**

Figure 11 Example of the price discrimination process.

In order to define a price discrimination process, two parameters need to be defined, the value of w and the amount of the increasing. The first parameter, w, needs to be a function of the number of job agents bidding for their last operation on each machine k. The largest the number of job agents bidding, the largest the value of w for that machine. The amount of the price increasing can be defined according to a non-linear function. Time slots closer to the CDD (i.e. those slots on the interval [CCD – w, CDD + w]) have a higher, value and the price decreases as time slots are more distant from the CDD.

Let FPT_k be

$$FPT_k = \sum_{\forall i \therefore m_{in_i} = k} p_{in_i} \qquad \textbf{Equation 45}$$

where FPT_k is the total processing time for all the final operations to be processed on machine k. Based on that, w_k can be defined as:

$$w_k = \lfloor q * FPT_k \rfloor$$ **Equation 46**

where q is an amplitude factor which is initially fixed at 0.2 (this value was obtained from the calibration process with the known optimal solutions of small problems), and $\lfloor x \rfloor$ means the integer lesser or equal to x.

Also, an exponential decay function shown in equation 46 is used to define the price increasing factor as follows:

$$\Delta_{kt} = 2e^{-0.007|CDD-t|}$$ **Equation 47**

If the time slot is close to the CDD (i.e. $t \approx CDD$), then the value of the increasing factor is 2, which means the price is doubled. As the time slot gets far from the CDD, the value of the factor tends to 1 (i.e. the price for that slot stays the same). This factor is applied on the interval [CCD – w_k, CDD + w_k]. Based on this price discrimination, an *augmented tâtonnement* utility function is defined.

Even with price discrimination, this price updating process might oscillate and not achieve convergence of the price vector. This is where the use of mathematical programming tools can help in designing a price adjustment scheme, where each iteration brings the prices closer to optimal allocation. In general, given a feasible schedule, the auctioneer would be able to calculate the optimal prices for each machine. However, this is not a trivial problem, and the step parameter s on equation 43 needs to satisfy certain

conditions in order to get a good convergence to the optimal Lagrangean dual prices, i.e. the optimal prices of the time slots for each machine. Fisher (1985) shows that an effective value is:

$$s_r = \alpha_r \left(\frac{UB - LB}{\sum_k \sum_t D_{kt}^2 (X_{ijt})} \right) \qquad \textbf{Equation 48}$$

where α_r is a scalar value satisfying $0 < \alpha_r \leq 2$, UB, and LB are upper and lower bounds to the problem, and D_{kt} is the excess demand. The lower bound can be calculated by adding all the job sub-problem optimal solutions. The upper bound can be calculated either by using the objective function value of the capacity feasible schedule or by using the optimal solution of the global problem. In the last case, the auction procedure reaches the equilibrium prices faster. Based on this price adjustment, an adaptive *tâtonnement* auction protocol is defined.

The algorithm can be summarized as follows: during the progress of the auction, each job agent i solves its locally constrained utility maximization problem to find the best combination of resource-time slots (B^*_i) given a resource price vector. All the job agents then submit their optimal bids to the auctioneer, who collects the new bids, computes and announces the updated resource prices, then proceeds with the next iteration. The optimal bids are $B^*_i = B^*_i(\lambda_{kt})$ computed from current machine time slot prices announced by the auctioneer.

Steps

1. Initialization: $r = 0$, $\lambda_{kt} = 0$, k= 1,..., m; t= 1,..., T; $\alpha_r = 2$.
2. With the prices (λ_{kt}) job agents solve their optimization problem. The solution is a job-level schedule.
3. The auctioneer combines all the bids and generates a capacity infeasible schedule. Objective function value plus total payments eventually to be done by job agents are equal to the lower bound (LB_r).
4. Capacity-feasible schedule done by the auctioneer (NP hard in general). Objective function value is equal to the upper bound (UB_r).
5. Updating the upper bound UB_r and α_r if necessary.
6. The auctioneer calculates the excess demand D_{kt} vector and updates the prices (λ_{kt}).
7. The auctioneer checks if the stopping criterion is satisfied. If not, the auctioneer starts the next iteration. Otherwise, he stops, and announces the best feasible schedule.

Example

In order to illustrate the proposed approach, the problem presented in Table 3 will be used. Recall that the problem had 2 machines, 5 jobs, and now the CDD is equal to 8, a restricted CDD. In this example, an adaptive price adjustment function, and a regular payment function will be used. A lower and upper bounds need to be defined in order to implement the adaptive version. Five job agents are created, the auctioneer will manage two machines ($m = 2$), and he will receive bids for 22 time slots ($T = 22$) on each

machine, 44 slots in total. T is defined based on an approximate value for the makespan, plus a value accounting for the processing time of the jobs expected to be late.

Initially, the iteration counter is set to zero, λ_{kt}^{0}, the prices vector is also set to zero, and α_0 is set to 2. Next, the 5 job agents solve their optimization problem. In this case, they try to minimize the earliness and tardiness cost, as well as the payment for the utilization of the two machines. Since the resource prices are all zero, only Jobs 4 and 5 will be late with one time unit each. Jobs 1, 2 and 3 will all be on time.

Figure 12 shows the capacity-infeasible schedule constructed by the auctioneer after the first iteration. As shown, while the job level schedules satisfy non-preemption and precedence constraints, the machine capacity constraints are violated in some time slots. Based on this schedule, the auctioneer calculates the lower bound LB_0, which is equal to 2 (earliness and tardiness penalties plus the payment for the two machines utilization).

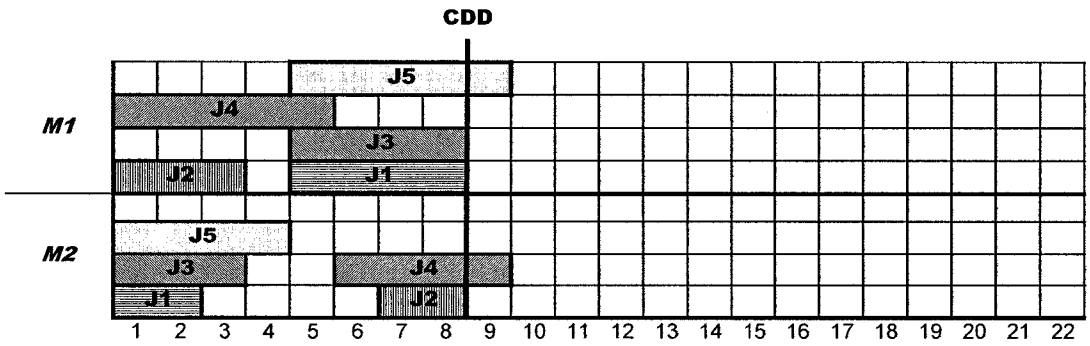


Figure 12 Capacity-infeasible schedule at first iteration.

Figure 13 shows the schedule with restored feasibility found by using the ranking procedure. Also, based on this restored schedule, the auctioneer calculates the upper bound UB_0, which is equal to 29.

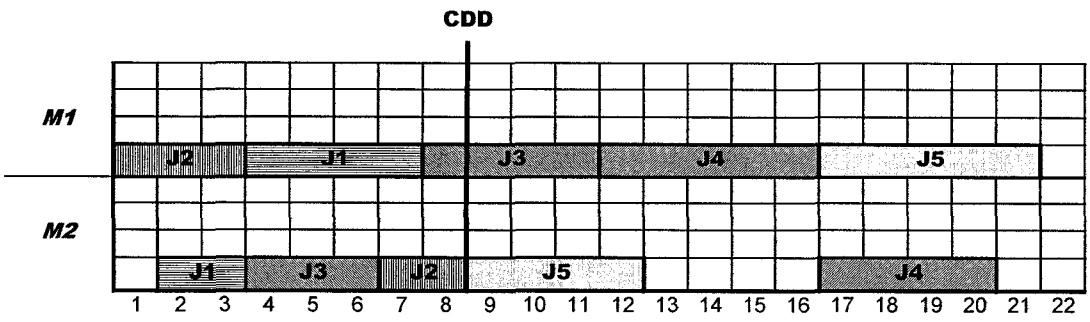
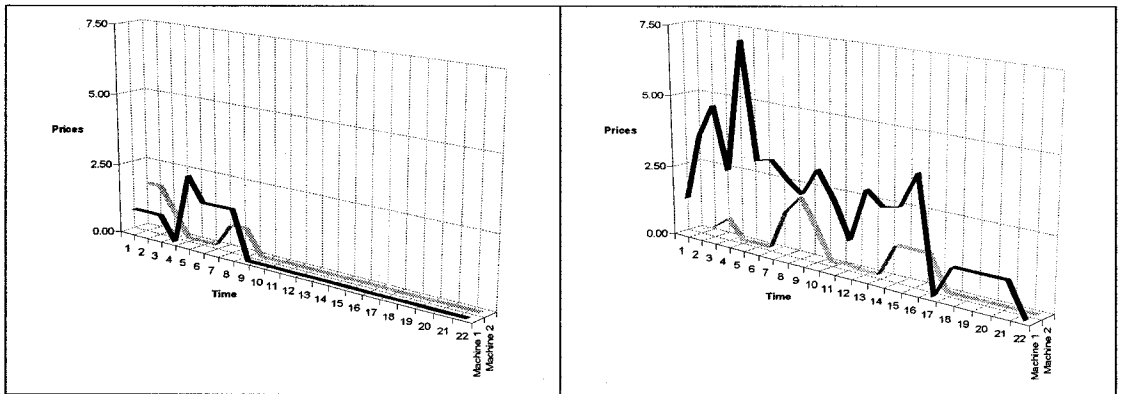


Figure 13 Capacity-feasible schedule.

Figure 14 shows the prices of the resources after the first iteration and at the final iteration. As shown in Figure 12, Jobs 1, 3, 4, and 5 are bidding for time slot 5 on Machine 1; Jobs 1, 3, and 5 are bidding for time slots 1 and 2 on Machine 2 and 6, 7, and 8 on Machine 1. Based on their demand, the auctioneer updates their prices, which consequently, are the highest among all the time slots. This procedure effectively captures the desirability of each time slot on each machine since the time slots that are highly demanded receive a higher value as shown in Figures 14a and 14b.



(a). Price Profiles at the end of the first iteration.

(b). Price Profiles at the end of the final iteration.

Figure 14 Price Profiles for the two machines.

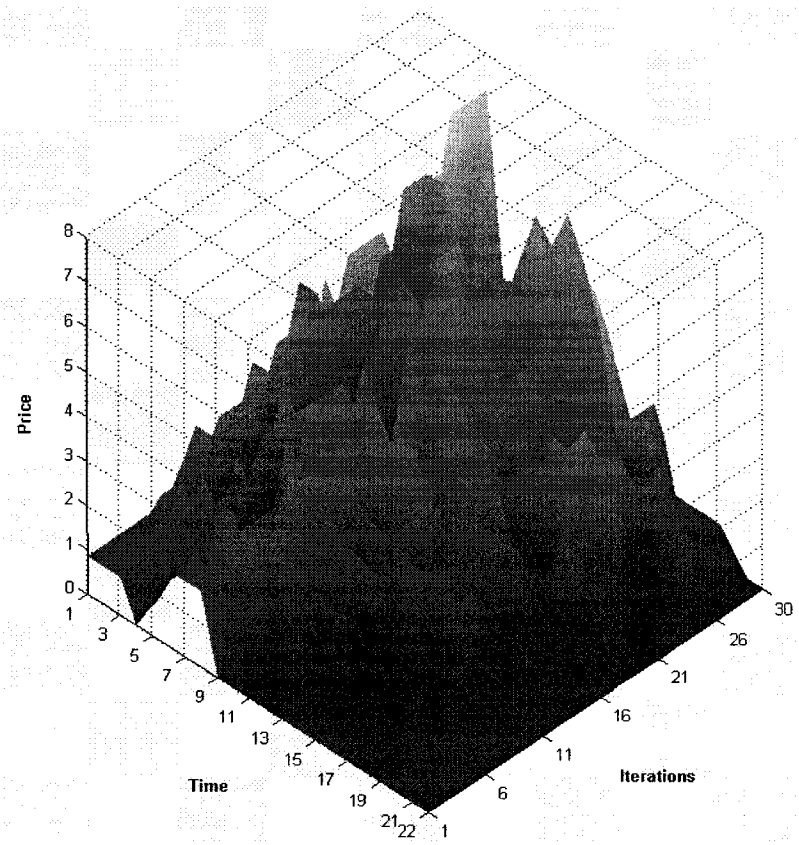


Figure 15 Price Profiles for Machine 1.

Figure 15 shows the evolution of prices for Machine 1. It can be noted how the auction procedure directs the job agents to bid for less attractive time slots, even if they are far from the CDD. At iteration 30, when the stopping criterion is reached, the price profile shows higher prices for time slots closer to the CDD. These time slots are highly demanded since the job agents are willing to pay for them to avoid tardiness cost.

Results

Sets of problems with 2, 3, 4, 5, 10, 15 and 20 machines; 5, 6, 7, 8, 9, 10, 20, 50, 100 and 500 jobs; with 30 problem instances per problem size were generated, in total 1800 problem instances were created. The processing times were generated from a discrete uniform distribution U(1, 10), and the job routes were obtained from another discrete uniform distribution U(1, m). Similar to most random number generators use today, the processing times and job routes were generated by using random numbers coming from a linear congruential generator (Law and Kelton 2000).

Although defining whether a CDD in a JSSP is restricted or unrestricted is a NP-Hard problem by itself (Lauff and Werner 2004a), based on the optimal makespan or the lower bound for each JSSP instance, its CDD can be generated. Taillard (1993) describes a procedure to calculate a lower bound for the makespan as a function of the parameters of the generated problem. Also, he conjectured that the lower bound found by using his procedure is tight enough if $(n/m) \rightarrow \infty$ and, therefore, can be used as an approximation of the optimal makespan.

After calculating the lower bound, the next step is to determine the CDD. Biskup and Feldmann (2001) proposed to use a more or less restricted CDD. A modification of their calculation can be used to generate each CDD as given in equation 49:

$$CDD = \lfloor h * LB \rfloor \qquad \textbf{Equation 49}$$

where h = 0.7, 0.8, 0.9, 0.95; LB is the lower bound as described by Taillard (1993), and again $\lfloor x \rfloor$ is the largest integer less than or equal to x. An unrestricted CDD for a *n*-jobs *m*-machines job shop scheduling problem can be calculated as the sum of all processing times on all machines (Lauff and Werner 2004a). Then, with this value, the earliness and tardiness cost for the unrestricted problem can be calculated by using the Kanet's algorithm from Kanet (1981) on each machine. This cost can be used either as a parameter to calculate the upper bound required in the auction procedure or as performance evaluation parameter.

Similar to the *n*-job 2-machins problem presented in Chapter 3, a Balance Ratio (BR) can be defined for larger problems. The BR for a *n*-jobs *m*-machines job shop scheduling problem can be defined as the ratio between the summations of the processing times of the operations to be performed on the machine, with the minimum summation and the machine with the maximum summation as shown in equation 50:

$$BR = \underset{\forall k}{Min}\left(\sum_{i=1}^{n} p_{ik}\right) / \underset{\forall k}{Max}\left(\sum_{i=1}^{n} p_{ik}\right) \qquad \textbf{Equation 50}$$

The closer BR is to one, the more balanced the problem is. In a balanced problem, the total amount of processing time on each machine is similar (i.e. there is no a clear bottleneck machine), and in theory, an even utilization of all the machines could be reached, then reducing the value of the objective function.

For problems with a tight CDD (i.e. the CDD as defined in equation 18), the more compact the schedule, the less the tardiness cost and, hence, the less the objective function value is. For balanced problems, the auction procedure is able to find equilibrium prices, and for small problems (2 machines with 5, 6, 7, and 8 jobs; 3 machines with 5 and 6 jobs), those prices correspond to the optimal dual solution when their known optimal solution is used as the upper bound in the auction procedure. Also, these optimal solutions were used to find a fit value of the parameter α_r in equation 47, which is required in the first step of the auction procedure. On the other hand, the auction procedure fails to find a feasible schedule with its corresponding equilibrium prices when the problem is less balanced (BR < 0.6), which accounts for 34% of the problems. For these problems, an *augmented tâtonnement* utility function was used as defined in equations 44 to 46. Once this utility function was used, the improved auction procedure was able to find a feasible schedule and its corresponding equilibrium prices for 75% of the problems. Also, because of computational limitations solutions for problems with 15 and 20 machines and 500 jobs were not found, these accounted for 3% of the problems.

Table 7 Performance of the auction procedure.

Machines	Jobs	CDD= 0.95*LB		CDD= 0.9*LB		CDD= 0.8*LB		CDD= 0.7*LB	
		AD	SD	AD	SD	AD	SD	AD	SD
2	9	39.0%	19.7%	39.0%	25.9%	50.0%	14.0%	60.0%	39.2%
2	10	25.0%	22.9%	47.0%	19.6%	46.0%	22.9%	62.0%	11.5%
2	20	36.0%	13.1%	46.0%	34.9%	48.0%	30.1%	59.0%	41.1%
2	50	40.0%	5.7%	49.0%	12.4%	46.0%	20.5%	51.0%	40.4%
2	100	39.0%	4.9%	40.0%	22.7%	56.0%	15.2%	63.0%	15.6%
2	500	40.0%	22.0%	49.0%	12.7%	53.0%	38.7%	66.0%	32.9%
3	7	38.0%	5.4%	45.0%	24.1%	50.0%	39.4%	72.0%	16.0%
3	8	36.0%	20.6%	39.0%	11.6%	60.0%	29.5%	62.0%	28.4%
3	9	39.0%	9.2%	39.0%	21.4%	45.0%	22.5%	70.0%	13.6%
3	10	31.0%	14.7%	35.0%	15.8%	55.0%	32.4%	82.0%	16.5%
3	20	36.0%	11.0%	37.0%	30.8%	52.0%	31.0%	84.0%	28.7%
3	50	38.0%	22.1%	42.0%	26.1%	49.0%	16.2%	84.0%	17.7%
3	100	30.0%	7.8%	50.0%	28.0%	49.0%	16.8%	51.0%	34.4%
3	500	29.0%	14.3%	50.0%	27.3%	50.0%	14.2%	79.0%	17.0%
4	5	33.0%	17.5%	42.0%	28.7%	56.0%	22.6%	51.0%	12.6%
4	6	34.0%	16.1%	38.0%	18.4%	55.0%	14.2%	69.0%	37.2%
4	7	33.0%	14.8%	44.0%	17.1%	47.0%	21.2%	69.0%	23.0%
4	8	33.0%	11.3%	37.0%	33.3%	56.0%	22.7%	81.0%	35.5%
4	9	34.0%	14.4%	43.0%	31.4%	60.0%	11.6%	54.0%	19.4%
4	10	38.0%	23.6%	35.0%	27.7%	54.0%	19.7%	50.0%	33.8%
4	20	40.0%	10.9%	39.0%	28.1%	54.0%	27.1%	56.0%	23.5%
4	50	26.0%	15.3%	35.0%	16.1%	56.0%	13.3%	85.0%	22.7%
4	100	33.0%	15.2%	38.0%	12.3%	51.0%	22.5%	51.0%	21.4%
4	500	27.0%	21.5%	50.0%	22.2%	56.0%	17.6%	82.0%	37.6%
5	5	28.0%	8.0%	45.0%	33.1%	46.0%	37.0%	57.0%	15.7%
5	6	27.0%	18.5%	49.0%	29.6%	58.0%	15.1%	70.0%	14.5%
5	7	31.0%	5.3%	43.0%	11.3%	48.0%	21.3%	84.0%	30.9%
5	8	34.0%	14.7%	44.0%	27.9%	56.0%	36.6%	87.0%	24.3%
5	9	31.0%	12.7%	35.0%	11.1%	45.0%	11.9%	53.0%	12.3%
5	10	28.0%	21.7%	44.0%	28.5%	51.0%	38.1%	56.0%	30.2%
5	20	36.0%	19.9%	38.0%	17.4%	47.0%	29.5%	72.0%	41.3%
5	50	28.0%	5.1%	43.0%	26.0%	48.0%	21.3%	66.0%	11.9%
5	100	29.0%	5.1%	42.0%	23.7%	48.0%	21.9%	73.0%	17.8%
5	500	26.0%	23.4%	43.0%	22.6%	50.0%	16.1%	71.0%	41.9%
10	5	39.0%	8.9%	47.0%	20.9%	52.0%	18.0%	70.0%	44.0%
10	6	25.0%	7.0%	38.0%	24.1%	60.0%	18.1%	63.0%	26.5%
10	7	37.0%	16.8%	49.0%	30.0%	45.0%	25.2%	64.0%	19.6%
10	8	33.0%	8.9%	43.0%	34.6%	56.0%	14.4%	60.0%	13.3%
10	9	28.0%	21.4%	50.0%	27.4%	46.0%	29.2%	72.0%	35.9%
10	10	37.0%	23.8%	36.0%	23.0%	60.0%	31.0%	53.0%	31.1%
10	20	37.0%	22.9%	42.0%	24.0%	45.0%	28.8%	64.0%	13.4%
10	50	28.0%	13.8%	50.0%	12.2%	45.0%	27.1%	59.0%	30.1%
10	100	30.0%	11.6%	41.0%	21.0%	58.0%	37.5%	70.0%	32.9%
10	500	39.0%	19.2%	49.0%	30.2%	46.0%	17.7%	51.0%	41.9%
15	5	32.0%	16.8%	49.0%	10.1%	48.0%	16.2%	66.0%	33.4%
15	6	25.0%	12.5%	37.0%	12.2%	50.0%	20.6%	52.0%	22.9%
15	7	26.0%	17.0%	41.0%	30.6%	51.0%	31.5%	78.0%	16.9%
15	8	32.0%	22.6%	41.0%	20.1%	52.0%	18.1%	66.0%	44.7%
15	9	26.0%	20.3%	35.0%	11.6%	50.0%	27.9%	78.0%	25.5%
15	10	34.0%	19.2%	38.0%	24.1%	57.0%	28.8%	62.0%	40.5%
15	20	33.0%	6.0%	45.0%	28.0%	51.0%	34.2%	70.0%	29.3%
15	50	25.0%	20.5%	40.0%	29.3%	60.0%	33.4%	85.0%	13.8%
15	100	32.0%	4.5%	47.0%	19.5%	59.0%	19.9%	56.0%	20.4%
15	500	-	-	-	-	-	-	-	-
20	5	31.0%	16.0%	37.0%	29.4%	54.0%	27.7%	61.0%	25.8%
20	6	40.0%	15.5%	38.0%	34.6%	60.0%	30.4%	69.0%	30.5%
20	7	37.0%	23.5%	48.0%	20.9%	45.0%	39.3%	52.0%	22.4%
20	8	36.0%	10.6%	37.0%	15.7%	58.0%	38.3%	71.0%	36.2%
20	9	38.0%	12.4%	47.0%	17.2%	47.0%	16.4%	63.0%	41.2%
20	10	39.0%	17.8%	48.0%	25.9%	55.0%	19.8%	61.0%	35.4%
20	20	36.0%	19.0%	50.0%	16.8%	52.0%	15.0%	57.0%	21.9%
20	50	39.0%	16.1%	41.0%	27.2%	48.0%	28.8%	81.0%	26.1%
20	100	30.0%	8.5%	36.0%	16.6%	52.0%	12.4%	84.0%	32.7%
20	500	-	-	-	-	-	-	-	-

For problems without optimal solution, the auction procedure is evaluated based on how far their solutions were from the earliness and tardiness cost for the unrestricted problem (*ETURP*). Recall that a problem instance's solution with a restricted CDD will always be larger than the same instance with unrestricted CDD. Table 7 shows the Average Deviation (*AD*) from the unrestricted version and its Standard Deviation (*SD*). *AD* is calculated as follows:

$$AD = [(\text{Auction procedure Solution} - ETURP)/ ETURP] \; X \; 100\%. \quad \textbf{Equation 51}$$

Note that regardless of the number of jobs and the number of machines, when the CDD is tighter in Table 7 (it gets tighter going from left to right), *AD* increases because of using the optimal solution of their equivalent unrestricted problem to evaluate their performance. *AD* does not seem to be affected when either the number of jobs or number of machines increases for the same tightness factor. In general, the bidding approach allows big problems to be decomposed in smaller problems and then be solved in parallel, and therefore, size (number of jobs and number of machines) does not seem to affect their performance. With better lower bounds or optimal solutions, a better result analysis can be carried out.

Figure 16 shows the behavior of *AD* for problems of different sizes. In general, regardless of the number of jobs and machines, *AD* increases as the CDD gets tighter. It does not change too much when the tightness factor decreases from 0.9 to 0.8, and in some cases, it even improves. However, when the factor decreases from 0.8 to 0.7, in most of the cases, there is a big increase in *AD*.

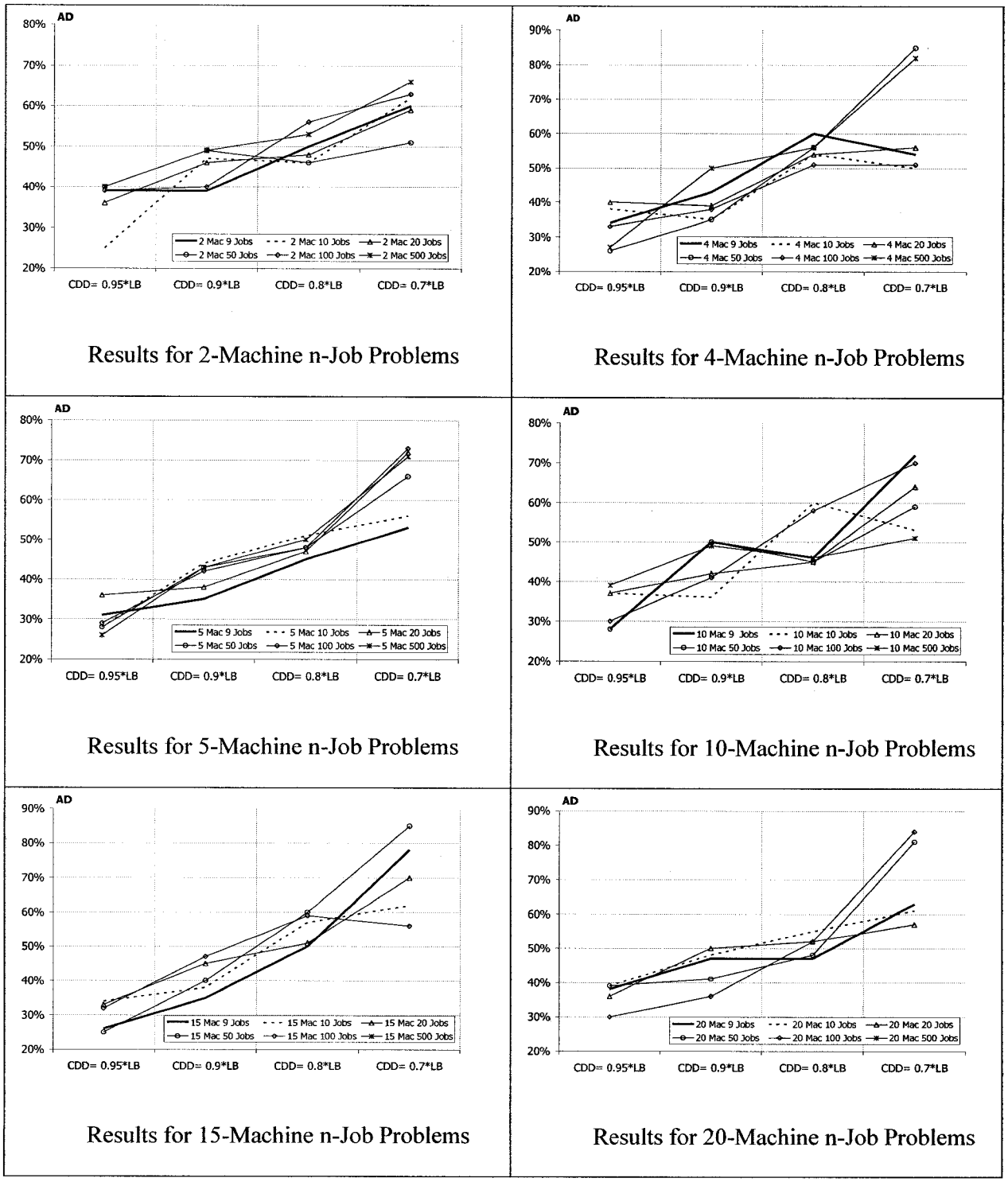


Figure 16 Results for *m*-machine *n*-job problems.

Comparison between the two heuristic methods for small problems

Generally speaking, most of the problems to be solved in scheduling are classified as NP-complete. That is, there are no efficient algorithms to solve a NP-complete problem in a time that is expressed as a polynomial or pseudo polynomial function of the size of the problem (Garey and Johnson 1979). In the literature, there are different approaches to measure the effectiveness of heuristic methods to solve NP-complete problems. For a heuristic procedure, it is important to evaluate how close the solution value is to the optimal value (Hall and Posner 2001), hence, finding the optimal values is decisive for this evaluation process. However, most of the time, optimal solutions are not available, and different approaches need to be followed.

In this research, optimal solutions for small (up to 8 jobs) two-machine JSSP E/T problems with a restricted CDD were obtained earlier and used here to evaluate the performance of the solutions of the two proposed heuristic methods. For balanced problems (i.e. those problems with a BR greater than or equal to 0.95 and less than or equal to 1.05), both the *JSSPET* algorithm and the auction-based approach were able to find optimal or close to optimal solutions (within a 5% deviation from the optimal solution). The auction-based approach used an adaptive *tâtonnement* protocol. In this approach, the known optimal solutions were used as the upper bound for the calculation of the step parameter in the adjustment of the time slots' prices during the iterative bidding process. This approach was able to find the equilibrium prices, i.e. the optimal solution for all the small problems and clearly outperforming the *JSSPET* algorithm.

Table 8 shows the Average Deviations (*AD*) from the optimal solutions and their Standard Deviations (*SD*) for the balanced problems per job size. *AD* is calculated as in Equation 51, where *Auction procedure Solution* is replaced by either the *JSSPET* algorithm or the auction-based approach results.

For unbalanced problems, on the other hand, neither the *JSSPET* algorithm nor the auction-based approach was able to find optimal or near optimal solutions when used with an adaptive *tâtonnement* protocol. In the case of the auction-based approach, equilibrium prices were not found for most of the unbalanced problems. However, when the auction-based approach used the price discrimination process with 100 iterations as a stopping criterion, and the known optimal solutions as an upper bound, equilibrium prices and their corresponding optimal resources allocation were found for all the problems outperforming the *JSSPET* algorithm. Table 9 shows the Average Deviations (*AD*) from the optimal solutions and their Standard Deviations (*SD*) for the unbalanced problems per job size.

It is clear that the auction-based approach did well because optimal solutions were used as an upper bound, and this will not be the case with large problems. Nevertheless, the benefit of using the small problems with known optimal solutions in this section is to calibrate the algorithms, i.e., to manually and/or automatically alter the value of the parameters in the auction-based approach in order to achieve a better performance for all the set of problems. The parameters to calibrate in the auction-based approach are q in

Equation 46, the rate in the exponential decay presented in Equation 47 and α_r in Equation 48.

Table 8 Performance of the auction procedure with small balanced problems.

Jobs	Balanced Problems Restricted CDD															
	$h = 0.70$				$h = 0.80$				$h = 0.90$				$h = 0.95$			
	JSSPET algorithm		Auction-based		*JSSPET* algorithm		Auction-based		*JSSPET* algorithm		Auction-based		*JSSPET* algorithm		Auction-based	
	AD	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*
n = 5	2.56%	1.64%	0.00%	0.00%	2.00%	1.35%	0.00%	0.00%	1.95%	1.35%	0.00%	0.00%	1.71%	1.38%	0.00%	0.00%
n = 6	2.63%	1.47%	0.00%	0.00%	2.19%	1.65%	0.00%	0.00%	2.19%	1.58%	0.00%	0.00%	1.59%	0.99%	0.00%	0.00%
n = 7	2.76%	0.99%	0.00%	0.00%	2.21%	1.51%	0.00%	0.00%	2.25%	1.28%	0.00%	0.00%	1.85%	1.42%	0.00%	0.00%
n = 8	2.88%	1.30%	0.00%	0.00%	2.44%	1.76%	0.00%	0.00%	2.27%	1.48%	0.00%	0.00%	2.01%	1.28%	0.00%	0.00%

Table 9 Performance of the auction procedure with small unbalanced problems.

Jobs	Unbalanced Problems Restricted CDD															
	$h = 0.70$				$h = 0.80$				$h = 0.90$				$h = 0.95$			
	JSSPET algorithm		Auction-based		*JSSPET* algorithm		Auction-based		*JSSPET* algorithm		Auction-based		*JSSPET* algorithm		Auction-based	
	AD	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*	*AD*	*SD*
n = 5	20.22%	3.74%	0.00%	0.00%	19.71%	2.92%	0.00%	0.00%	20.77%	2.58%	0.00%	0.00%	20.36%	3.24%	0.00%	0.00%
n = 6	21.21%	3.34%	0.00%	0.00%	19.15%	2.86%	0.00%	0.00%	19.26%	3.40%	0.00%	0.00%	19.78%	3.24%	0.00%	0.00%
n = 7	21.07%	3.63%	0.00%	0.00%	20.18%	3.02%	0.00%	0.00%	20.16%	2.07%	0.00%	0.00%	18.85%	3.16%	0.00%	0.00%
n = 8	22.66%	2.88%	0.00%	0.00%	20.25%	3.08%	0.00%	0.00%	20.77%	2.74%	0.00%	0.00%	19.82%	3.70%	0.00%	0.00%

CHAPTER V.

CONCLUSIONS AND FURTHER RESEARCH

For many years, scheduling research focused on regular measures of performance. Just recently, earliness and tardiness problems have become important with the new emphasis on just-in-time (JIT) production. Because of that, scheduling problems for meeting due date requirements have been widely studied, and among such problems are the scheduling problems involving earliness and tardiness (E/T) penalties over a common due date (CDD). In this research, exact and heuristic methods for solving the job shop scheduling problem (JSSP) considering earliness and tardiness over a common due date are presented. Throughout this dissertation, this problem is referred as JSSP E/T, and as far as I know, there is not research addressing this problem. In this research, both exact and heuristic algorithms are developed to find optimal or near optimal solutions for this problem.

This chapter is divided into three sections; first, the contribution of this research is introduced. Next, the conclusions of the exact and heuristic methods developed are presented, and finally, the future research is discussed in the last section.

Contributions

The contributions of the proposed research can be summarized as follows:

- A theoretical contribution is achieved by introducing an analytical approach to deal with multi-stages scheduling problems involving non-regular measures of performance. The JSSP E/T with restricted CDD, as a multi-stage scheduling problem, is known to be NP-Hard in the strong sense.
- A polynomial procedure to determine the restrictedness of the CDD for a two-machine JSSP E/T, which allows classifying these problems as restricted, semi-restricted, and unrestricted based on their processing times and how large the CDD is.
- Properties of the JSSP E/T with restricted CDD, as well as some optimality conditions are derived and are used to derive heuristic methods to solve this problem.
- The two-machine n-job problem is studied initially as a good point of reference to extend the findings to the m-machine, n-job environment. Optimal solutions for some cases of the two-machine, n-job problem are derived by using dynamic programming.
- An innovative application of Multi-Agent Systems (MAS) using bidding with combinational auctions is developed to solve multi-stage scheduling problems considering earliness and tardiness as a measure of performance.
- A set of benchmark problems is created in order to evaluate the performance of the proposed exact and heuristic methods. These problems will be useful to the scheduling community to evaluate different approaches to solve the JSSP E/T considering a CDD.

Conclusions

Initially, for a two-machine job shop scheduling problem, the CDD is classified as unrestricted, restricted and semi-restricted, depending on how large it is. Lauff and Werner (2004[b]) conjectured that the definition of the restrictedness of the CDD for a multi-machine JSSP E/T is a NP-Hard problem. Here, a polynomial procedure to define the class of restrictedness for the two machines JSSP E/T over a CDD was presented.

Additionally, some properties for this problem, as well as optimality conditions for the unrestricted and semi-restricted case, were extended from the single machine problem. Optimal solutions for the unrestricted and semi-restricted case were obtained for problems with up to 500 jobs by using dynamic programming. Two properties for the restricted case were proved and used to come up with a heuristic algorithm for the two machines problem with restricted CDD.

Furthermore, through experiments, it has been shown that the developed algorithm for the two-machine problem can find optimal solutions for the unrestricted and semi-restricted version of the problem. Also, the algorithm works well as a heuristic for the restricted case with large problems.

Regarding heuristic methods, price-directed auction mechanisms for distributed scheduling were introduced for the JSSP E/T over a CDD. Two auction protocols (non-adaptive Walrasian and adaptive tâtonnement) and two payment functions (regular and augmented tâtonnement) were investigated. Based on previous research, the well-known Lagrangean Relaxation for resource allocation problems using sub gradient search was used as a way to implement combinational auctions for resource allocation. A large set of JSSP E/T instances over a CDD was solved using the proposed approach. As

demonstrated in the computational experiments, the prices of time slots (objects) depend heavily on the demand patterns of job agents which, in turn, are based on their objective function value (i.e., minimizing earliness and tardiness costs over a CDD).

Through a set of properties extended from the two-machine to the *m*-machine problem it was shown that, in general, the optimal schedule is V-shaped (strongly or weakly v-shaped). For the restricted case, this property holds for a subset of jobs, not for all jobs, even in the two-machine problem. As the CDD gets tighter, the number of jobs in this subset decreases as more jobs will be tardy. In such a case, it is better to schedule jobs with small total processing in all the machines around the CDD as the remaining jobs will be late anyway. Based on my observations, most optimal (or high quality) solutions exhibit a V-shape behavior; however, in a few cases, this pattern may not be easily detected and half V-shapes may exit.

Also, in this research, some recent questions concerning distributed scheduling were explored. By investigating different alternatives for implementation and their possible implications, a wide set of problems was solved and interesting insights found. The computational experiments demonstrated that the adaptive price update with sub-gradient step might be superior to the non-adaptive auction protocol. Moreover, augmented tâtonnement using a price discrimination scheme was effective in speeding up the convergence and in finding equilibrium prices for unbalanced problems (problems with bottleneck machines).

Additionally, the experiments demonstrated that when known optimal solutions are used as upper bounds in the auction procedure, it is possible to find the equilibrium prices faster. Also, parameters for the adaptive price update were better calibrated when

using known optimal solutions for small problems; then, their best values were used for solving larger problems. In this way, when those larger problems were solved with the adaptive price updating and with the price discrimination scheme, the algorithm was able to find the equilibrium prices for both balanced and unbalanced problems. However, because of the lack of optimal solutions or good lower bounds for larger problems, this approach does not seem to perform that well (See Table 7). Although the existence of equilibrium prices does not guarantee an optimal allocation of resources in a general combinational auction (Wellman et al 2001), the fact that the auction-based approach was able to find those prices is a good indication of the potential of this approach.

A very clear and valuable finding of this research is that when optimal solutions were available or lower bounds for those solutions were used, the computational experiments showed that the heuristic algorithms proposed in this research work well when the problem is balanced (i.e. there is no bottleneck machine). In this sense, the balance ratio defined in this research seems to be a good predictor of the quality of the solution when it is found by using the proposed heuristic algorithms. In other words, if the data for a certain instance shows that the problem will be balanced, then it is expected that the proposed algorithm will perform well; otherwise, the algorithm does not guarantee the quality of the solution.

Future Research

This research has defined an interesting scheduling problem involving a non-regular measure of performance and has opened some new research problems. By defining a new problem and extending existing methods for solving it, both exact and heuristic methods can be developed. In particular, the following points are given as follow-up research directions:

1. Extend the definition of the restrictedness of the CDD for a multi-machine environment with more than two machines: By developing properties for a multi-machine JSSP E/T, the nature of the restrictedness of the CDD can be defined.
2. Develop optimality conditions and a dynamic programming algorithm(s) to solve the restricted case of the two-machine JSSP E/T over a CDD: Some properties and optimality conditions were developed for the restricted case, but there is still room to develop an exact algorithm to find optimal solutions. In particular, an extension of the dynamic programming approach proposed in this research is a first step in this direction.
3. Improve the performance of the *JSSPET* algorithm by introducing better lower bounds for the two-machine E/T JSSP over a restricted CDD to better evaluate the performance of the heuristic: By developing better lower bounds, not only for the proposed heuristic approaches, but also for any new heuristic, their performance can be evaluated in a more accurate

way. Further, based on the new, improved lower bounds, information to find the "best" heuristic approach can be obtained.

4. Develop new heuristic methods for the multi-machine JSSP E/T over a restricted CDD: Novel heuristic methods, like Tabu Search (TS) and Simulated Annealing (SA), can be used as well to find solutions for the problem addressed in this research. Furthermore, a comparison of the performance among all proposed heuristic could be conducted by using the set of benchmark problems proposed in this research.

5. Extend the auction-based approach from the static scheduling problems to a dynamic environment: Many real problems can be formulated as a job shop scheduling problem in a dynamic environment. In such an environment, jobs arrive to the shop by following a random distribution, so the scheduling process needs to be re-formulated every time a new job arrives. Since this approach shows a good performance, it can be extended without major variation to problems involving the same measure of performance in a dynamic environment.

6. Find optimal solutions or good lower bounds for large problems: Either optimal solutions or good lower bounds are decisive for the performance evaluation not only of the two heuristic methods proposed in this research, but also for future heuristics methods developed to deal with the JSSP E/T over a CDD.

7. Implement a different strategy of distributed control: Bidding with combinational auctions was successfully implemented in this research for

JSSPs. However, other strategies like negotiation and cooperation may be implemented and evaluated against the bidding strategy.

8. Relax some of the assumptions of the problem: In this research, some assumptions regarding the JSSP E/T were made. Specially, α_i and β_i were assumed to be equal to 1. In a more realistic environment, jobs might have different priorities, and therefore, their penalty function should vary accordingly. Also, earliness or tardiness might have different cost penalties based on the nature of the jobs and the importance of the customer. Therefore, having different values for them, will be an interesting, but more difficult problem to solve. Finally, rather than having common due date, a common due window within which no penalties are incurred, might be a more realistic assumption.

REFERENCES

Adams J., Balas E., and Zawack D., (1988). The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, 34, 391-401.

Ahmadi R. H. and Bagchi U., (1990). Lower Bounds for Single-Machine Scheduling Problems. *Naval Research Logistics*, **37**, 967-979.

Aydin M. E. and Oztemel E., (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, **33**, 169-178.

Bagchi U., Sullivan R. S., and Chang Y. L., (1986). Minimizing Mean Absolute Deviation of Completion Times about a Common Due Date. *Naval Research Logistics Quarterly*, **33**, 227-240.

Baker K. R., (1997). *Elements of sequencing and scheduling.* Kenneth Baker Editors.

Baker K. R. and Scudder G. D., (1990). Sequencing with earliness and tardiness penalties: A review. *Operations Research*, **38**, 22-36.

Bank J. and Werner F., (2001). Heuristic algorithms for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties. *Mathematical and Computer Modelling*, **33**, 363-383.

Bedoya L. and Rabadi G., (2005). Two machines job shop scheduling problem considering earliness and tardiness with a common due date. Working Paper, Old Dominion University.

Bertsekas D., "Ten simple rules for mathematical writing", (http://web.mit.edu/dimitrib/www/home.html visited 01/25/04).

Bertsekas D, (1988). The auction algorithm: a distributed relaxation method for the assignment problem. *Annals of Operations Research*, **14**, 105-123.

Biskup D. and Feldmann M., (2001). Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due date. *Computers and Operations Research*, **28**, 787-801.

Biskup D. and Simons D., (1999). Game theoretic approaches to cost allocation in the dynamic total tardiness problem. *IIE Transactions*, **31**, 899-908.

Chang S., Matsuo H., and Tang G., (1990). Worst-Case Analysis of Local Search Heuristics for the One-Machine Total Tardiness Problem. *Naval Research Logistics*, **37**, 111-121.

Chen B., (1994). *Worst-Case Performance of Scheduling Heuristics*. Thesis Publishers Amsterdam.

Cheng T. C. E. and Podolsky S., (1996). *Just in Time Manufacturing. An introduction.* Chapman & Hall, Second Edition.

Conway R. W., Maxwell W. L., and Miller W. L., (1967). *Theory of Scheduling.* Addison-Wesley, Reading, MA.

Crowe T. J. and Stahlman E. J., (1995). A proposed structure for distributed shopfloor control. *Integrated Manufacturing Systems*, **6**, 31-36.

Davis R. and Smith R. G., (1983). Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, **20**, 63-109.

Dang T. T. and Frankovic B., (2002). Agent-based scheduling in production systems. *International Journal of Production Research*, **40**, 3669-3679.

Danneberg D., Tautenhahn T., and Werner F., (1999). A Comparison of Heuristic Algorithms for Flow Shop Scheduling Problems with Setup Times and Limited Batch Size. *Mathematical and Computer Modelling*, **29**, 101-126.

Decker K. S., (1987). Distributed Problem-Solving Techniques: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC 17**, 729-740.

Demange G., Gale D., and Sotomayor M., (1986). Multi-Item Auctions. *The Journal of Political Economy*, **94**, 863-872.

Demirkol E., Mehta S., and Uzsoy R., (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research*, **109**, 137-141.

Dewan P. and Joshi S., (2002). Auction-based distributed scheduling in a dynamic job shop environment. *International Journal of Production Research*, **40**, 1173-1191.

Emmons H., (1987). Scheduling to a common due date on parallel uniform processors. *Naval Research Logistics*, **34**, 803-810.

Fabiunake M. and Kock, G., (2000). A connectionist method to solve job shop problems. *Cybernetics and Systems – An international Journal*, **31**, 491-506.

Federgruen A. and Mosheiov, G., (1996). Heuristics for multimachine scheduling problems with earliness and tardiness costs. *Management Science*, **42**, 1544-1555.

Feldmann M. and Biskup D., (2003). Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers and Industrial Engineering*, **44**, 307-323.

Fisher M., (1981). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, **27**, 1-18.

Fisher M., (1985). An Applications Oriented Guide to Lagrangian Relaxation. *Interfaces*, **15**, 74-80.

Garey M. R. and Johnson D. S., (1979). *Computers and Intractability. A guide to the theory of NP-Completeness*. W. H. Freeman and Company, San Francisco.

Garey M. R., Johnson D. S., and Sethi R., (1976). The complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, **1**, 117-129.

Gordon V., Proth J. M. and Chu C., (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, **139**, 1-25.

Graham R. L., (1966). Bounds for certain multiprocessing timing anomalies. *Bell System Technical Journal*, **45**, 1563-1581.

Gupta J. N. D., Hennig K., and Werner F., (2002). Local search heuristics for two-stage flow shop problems with secondary criterion. *Computers & Operations Research*, **29**, 123-149.

Gupta J. N. D., Lauff V., and Werner F., (2004). Two-MachineFlow Shop Scheduling with Nonregular Criteria. *Journal of Mathematical Modelling and Algorithms*, **3**, 123-151.

Hall N. G., Kubiak W., and Sethi S. P., (1991). Earliness-Tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. *Operations Research*, **39**, 847-856.

Hall N. G. and Posner M. E., (1991). Earliness-Tardiness scheduling problems, I: Weighted Deviation of completion times about a common due date. *Operations Research*, **39**, 836-846.

Hall N. G. and Posner M. E., (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, **49**, 854-865.

Hino C. M., Ronconi D. P., and Mendes A. B., (2005). Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research*, **160**, 190-201.

Hoogeveen J. A. and Van de Velde S. L., (1991). Scheduling around a small common due date. European Journal of Operational Research, **55**, pp. 237-242.

Jozefowska J., Mik, M., Royck, R., Waligora G., and Wglarz J. W. (1998) Local search meta-heuristics for discrete-continuous scheduling problems. *European Journal of Operational Research*, V. 107, pp. 354-370.

Kanet J. J., (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly*, **28**, 643-651.

Kutanoglu E. and Wu D., (1999). On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. *IIE Transactions*, **31**, 813-826.

Lauff V. and Werner F., (2004a). On the complexity and some properties of multi-stage scheduling problems with earliness and tardiness penalties. *Computers and Operations Research*, **31**, 317-345.

Lauff V. and Werner F., (2004b). Scheduling with a common due date, earliness and tardiness penalties for multimachine problems: A survey. *Mathematical and Computer Modelling*, **40**, 637-655.

Lauff V. and Werner F., (2004c). Heuristics for two-machine flow shop problems with earliness and tardiness penalties. *International Journal of Operations and Quantitative Management*, **10**, 125-144.

Law A. M. and Kelton W. D., (2000). *Simulation Modeling and Analysis*. McGraw Hill, Boston, Third Edition.

Lee, I. (2001) Artificial intelligence search methods for multi-machine two-stage scheduling with due date penalty, inventory, and machining costs. *Computers and Operations Research*, V. 28, pp. 835-852.

Lin G. Y. and Solberg J. J., (1992). Integrated Shop Floor Control using Autonomous Agents. *IIE Transactions*, **24**, 57-71.

Macchiaroli R. and Riemma S., (2002). A negotiation scheme for autonomous agents in job shop scheduling. *International Journal of Computer Integrated Manufacturing*, **15**, 222-232.

Pinedo M., (1995). *Scheduling. Theory, Algorithms and, Systems.* Prentice Hall, New York.

Pinedo M., (2002). *Scheduling. Theory, Algorithms and, Systems.* Second Edition, Prentice Hall, Upper Saddle River, New Jersey.

Pinedo M. and Chao X., (1999). *Operations Scheduling with applications in Manufacturing and Services.* McGraw Hill.

Pritsker A., Watters L., and Wolfe P., (1969). Multiproject scheduling with limited resources: A zero–one programming approach. *Management Science*, **16**, 93–107.

Rabadi G., Mollaghasemi M., and Anagnostopoulos G. C., (2004). A branch-and-bound algorithm for the early/tardy machine-scheduling problem with a common due-date and sequence-dependent setup time. *Computers & Operations Research*, **31**, 1727-1751.

Rabadi G., Anagnostopoulos G., and Mollaghasemi M. (2007). A Heuristic Algorithm for The Just-In-Time Single Machine Scheduling Problem With Setups: A Comparison With A Simulated Annealing. *The International Journal of Advanced Manufacturing Technology*, **32**, 326–335.

Raghavachari M., (1986). A V-shape property of optimal schedule of jobs about a common due date. *European Journal of Operational Research*, **23**, 401-402.

Reeves M. D., Wellman M. P., Mackie-Mason J. K., and Osepayshvili A., (2005). Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, **39**, 67–85.

Rothkopf M. H., Pekec A., and Harstad R. M. (1998). Computationally Manageable Combinational Auctions. Management Science, 44, 1131–1147.

Sabuncuoglu I. and Toptal A., (1999a). Distributed scheduling: Part 1 – A review of concepts and applications. Technical paper: IEOR-9910, Bilkent University, Turkey.

Sabuncuoglu I. and Toptal A., (1999b). Distributed scheduling: Part 2 – Bidding algorithms and comparisons. Technical paper: IEOR-9911, Bilkent University, Turkey.

Sabuncuoglu I. and Toptal A., (1999c). Team-based Algorithms for Distributed Scheduling. Technical paper: IEOR-9912, Bilkent University, Turkey.

Sarper H., (1995). Minimizing the sum of absolute deviations about a common due date for two-machine flow shop problem. *Applied Mathematical Modelling*, **19**, 153-161.

Schniederjans M. J. and Olson J. R., (1999). *Advanced Topics in Just in Time Management*, Quorum Books.

Sousa P. and Ramos C., (1999). A distributed architecture and negotiation protocol for scheduling in manufacturing systems. *Computers in Industry*, **38**, 103-113.

Stevenson W. J., (1999). *Production Operations Management.* McGraw Hill, USA. Sixth Edition.

Sun H. and Wang G., (2003). Parallel machine earliness and tardiness scheduling with proportional weights. *Computers and Operations Research*, **30**, 801–808.

Sundararaghavan P. S. and Ahmed M. U., (1984). Minimizing the sum of absolute lateness in single-machine and multimachine scheduling. Naval research Logistics Quarterly, **31**, 325-333.

Sung C. S. and Min J. I., (2001). Scheduling in a two-machine flowshop with a batch processing machine(s) for earliness/tardiness measure under a common due date. *European Journal of Operational Research*, **131**, 95-106.

Szwarc W., (1989). Single-Machine Scheduling to Minimize Absolute Deviation of Completion Times from a Common Due Date. *Naval Research Logistics*, **36**, 663-673.

Taillard E. D., (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, **64**, 278-285.

Taillard E. D., (1994). Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *ORSA Journal on Computing*, **6**, 108-117.

T'kindt V. and Billaut J. C., (2002). *Multicriteria scheduling. Theory, Models and Algorithms*. Springer Verlag.

Wellman M. P., Walsh W. E., Wurman P. R., and Mackie-Mason J. K., (2001). Auction Protocols for Decentralized Scheduling. *Games and Economic Behavior*, **35**, 271–303.

Wellner J. and Dilger W., (1999). Job Shop with Multiagents. 13th Workshop 'Planen und Konfigurieren' (PuK) University of Würzburg.

Wooldridge, M. J., (2002). *An introduction to multiagent systems*. John Wiley & Sons Ltd.

Wu Z. and Weng M. X., (2005). Multiagent Scheduling Method with Earliness and Tardiness Objectives in Flexible Job Shops. *IEEE Transactions on Systems, Man, and Cybernetics –Part B: Cybernetics*, **35**, 293-301.

Zegordi S. H, Itoh K. and Enkawa, T., (1995). A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling problem. *International Journal of Production Research*, **33**, 1449-1466.

Youssef, H., Sait, S. M. and Adiche, H. (2001) Evolutionary algorithms, simulated annealing and tabu search: a comparative study. *Engineering Applications of Artificial Intelligence*, V. 14 (2), pp. 167-181.