

DISTRIBUTED RESOURCE SCHEDULING:
OPTIMIZATION MODELS, EQUILIBRIUM CONDITIONS,
AND INCENTIVE COMPATIBLE MECHANISMS

by
Erhan Kutanoglu

Presented to the Graduate and Research Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Industrial Engineering

Lehigh University
February, 1999

Approved and recommended for acceptance as a dissertation in partial fulfillment of
the requirements for the degree of Doctor of Philosophy.

(Date)

(Accepted Date)

S. David Wu, Ph.D.
(Dissertation Advisor)

Patrick T. Harker, Ph.D.

Louis A. Martin-Vega. Ph.D., P.E.

Robert H. Storer, Ph.D.

George R. Wilson, Ph.D.

To my wife Oya and my daughter Selin

Acknowledgements

First of all, I would like to thank to my advisor, Dr. David Wu, for his supervision, patient guidance, and encouragement throughout my Ph.D. study during the past four years. His continuous support and understanding has made this work possible.

I would like to express my special thanks to Dr. Pat Harker for his frequent comments about my research work in the Lehigh-Penn joint meetings.

I also wish to thank to the other members of my Ph.D. committee, Dr. Louis Martin-Vega, Dr. Bob Storer and Dr. George Wilson for their careful analysis of my proposal and initial manuscript of my dissertation, and for their suggestions that improved both.

I also thank to my friends who were or have been members of the Manufacturing Logistics Institute at Lehigh during the last four years for contributing to a warm and constructive discussion environment that I have been in.

I want to express my gratitude, love, and thanks to my wife Oya for her unconditional support and encouragement. Without her at my side, this would not have been possible. Finally, my special thanks go to my daughter Selin for her unlimited energy and wish to play with me which provided me the best meditation when I need it.

Contents

Acknowledgements	iv
Abstract	1
1 Introduction	2
1.1 Background and Motivation	2
1.2 Dissertation Outline	5
2 Combinatorial Auction for Scheduling	9
2.1 Introduction	9
2.2 A Survey of Related Literature	11
2.2.1 Distributed and Agent-based Scheduling	11
2.2.2 Lagrangean Relaxation Techniques	12
2.2.3 Auction Mechanism Design and Auction Algorithms	14
2.3 Problem Statement and Solution	
Methodology	16
2.3.1 Problem Statement	16
2.3.2 A Combinatorial Auction Mechanism for Scheduling	16
2.3.3 A Game Theoretic Perspective	21
2.3.4 Lagrangean Relaxation	22
2.3.5 Combinatorial Auction and Lagrangean Relaxation	25
2.4 Examples	30
2.4.1 Random job shop	32
2.4.2 Bottleneck Job Shop	36

2.4.3	Flow Shop	37
2.5	Conclusions	42
3	Robustness in Distributed Scheduling	43
3.1	Introduction	43
3.2	Problem Statement and Analysis	45
3.2.1	The Mathematical Model	45
3.2.2	The Proposed Scheduling Scheme and Broader Managerial Insights	47
3.3	Solution Methodology	49
3.3.1	Lagrangian Relaxation	50
3.3.2	Chance-Constrained Programming	51
3.3.3	Scaling, Lagrangian Ranking and Schedule Preprocessing	54
3.3.4	The Subgradient Search Algorithm	56
3.4	Computational Study	58
3.4.1	Experimental Design	58
3.4.2	The Dynamic Dispatching Benchmark	60
3.4.3	Testing Implementation Choices	61
3.4.4	Computational Results	62
3.5	Conclusions	74
4	Proactive Equilibrium for Distributed Scheduling	76
4.1	Introduction	76
4.2	Proactive Equilibrium	78
4.3	The Job Shop Scheduling Model	80
4.4	Proactive Equilibrium for JSP	84
4.5	Model of Others	86
4.5.1	Rank Based Precedence Constraints	87
4.5.2	Special Case: 1 out 1 constraints	93
4.5.3	Properties of the Job Agent Subproblem with 1 out of K Con- straints	94

4.5.4	CEST Constraints	95
4.6	An Algorithm for Proactive Equilibrium for JSP	98
4.7	An Illustrative Example	101
4.8	Conclusions	108
5	Incentive Compatible Scheduling	109
5.1	Introduction	109
5.2	The Schedule Selection Game	111
5.2.1	Problem Analysis	112
5.2.2	Dominant Strategy Equilibrium	115
5.2.3	Balancing the Budget	119
5.2.4	Participation and Asynchronous Decision Making	121
5.3	An Industry Case and Illustrative Example	124
5.3.1	Background	124
5.3.2	A Numerical Example	127
5.4	Discussion	131
5.4.1	Generation of Schedules	131
5.4.2	Aligning Original Performance and Transfers	133
5.4.3	Coordinated Scheduling with a Central Authority	134
5.5	Conclusions	136
6	Conclusions	137
6.1	Main Contributions	137
	Bibliography	140
	Vita	151

List of Tables

2.1	Notation	17
2.2	3x3 Random job shop example	32
2.3	3x3 Bottleneck job shop example	36
2.4	3x3 Flow shop example	40
3.1	Notation	45
3.2	Experimental factors and their levels in the simulation experiments .	59
3.3	Experimental results with ATC dispatching	60
3.4	Experimental results of the PFSL methodology	61
4.1	Notation	81
4.2	Data for the example: Single machine, 33 jobs	102
5.1	An example table for the Groves-Clarke mechanism for 8 schedules of 33 jobs	130

List of Figures

2.1	An iterative combinatorial auction mechanism for resource scheduling	31
2.2	Infeasible shop schedule after combining the job bids (first iteration) .	33
2.3	Price profiles at the end of <i>adaptive regular tâtonnement</i>	34
2.4	Feasibility-restored shop schedule after the fourth iteration (the best UB schedule which is also optimal).	34
2.5	Lower bound (LB) and Upper bound (UB) during the <i>adaptive regular tâtonnement</i>	35
2.6	Lower bound (LB) and Upper bound (UB) values during the <i>non- adaptive regular tâtonnement</i>	35
2.7	Lower bound (LB) and Upper bound (UB) values during the <i>adaptive augmented tâtonnement</i>	36
2.8	Lower bound (LB) and Upper bound (UB) during the <i>non-adaptive augmented tâtonnement</i>	37
2.9	Price profiles for each machine at the end of <i>adaptive regular tâtonnement</i> (Bottleneck Job Shop)	38
2.10	Lower bound (LB) and Upper bound (UB) values during the <i>adaptive regular tâtonnement</i> (Bottleneck Job Shop)	38
2.11	Lower bound (LB) and Upper bound (UB) values during the <i>non- adaptive regular tâtonnement</i> (Bottleneck Job Shop)	39
2.12	Lower bound (LB) and Upper bound (UB) values in <i>adaptive aug- mented tâtonnement</i> (Bottleneck Job Shop)	39
2.13	Price profiles at the end of <i>adaptive regular tâtonnement</i> (Flow Shop)	40

2.14	Lower bound (LB) and Upper bound (UB) values during the <i>adaptive regular tâtonnement</i> (Flow Shop)	41
2.15	Lower bound (LB) and Upper bound (UB) during <i>adaptive augmented tâtonnement</i> (Flow shop)	41
3.1	The Subgradient Search Algorithm	57
3.2	Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in <i>Uniform</i> processing time variation case (10x10 problems) .	65
3.3	Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in <i>Normal</i> processing time variation case (10x10 problems) .	66
3.4	Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in <i>Uniform</i> processing time variation case (20x15 problems) .	67
3.5	Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in <i>Normal</i> processing time variation case (20x15 problems) .	68
3.6	Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in <i>Uniform</i> processing time variation case (30x10 problems) .	69
3.7	Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in <i>Normal</i> processing time variation case (30x10 problems) .	70
3.8	Effects of surrogate scheme ($\Delta = 100$) on PFSL algorithm. (Uniform processing time variation case, 20x15 problems)	72
3.9	Effects of surrogate scheme ($\Delta = 100$) on PFSL algorithm. (Normal processing time variation case, 20x15 problems)	73
4.1	Lower bounds computed by using the base model and the model with the CEST/ranking-based precedence constraints	106
4.2	Upper bounds computed by using the base model and the model with the CEST/ranking-based precedence constraints	107

Abstract

The goal of this research is to establish a theoretical base for distributed resource scheduling. Our intention is to map optimization-based scheduling models with equilibrium seeking multiple agent models so that significant insights can be gained for the design and implementation of distributed scheduling systems. We develop models and tools that incorporate the distributed decision structure of manufacturing environments. The main challenge posed by these systems is to coordinate local interests and objectives while achieving globally sound decisions. A key requirement is to maintain private information and to allow local preferences to prevail. We first address this issue by introducing an iterative price-directed auction mechanism where job agents act on their own behalf and compete for shared resources. We show the connection between such auction mechanism and Lagrangean-based decomposition. We next address the issues of communication overhead and solution feasibility in such environment by introducing the concept of proactive equilibrium. Proactive equilibrium provides a basis that connects the theoretical underpinning between optimization and multiple agent models. We finally address the issue of incentive compatibility in the context of a schedule selection game. We propose a direct revelation mechanism that implements an optimal schedule under dominant strategies. We support and illustrate different segments of our conclusions through computational testing or industry case examples constructed from real life scenarios.

Chapter 1

Introduction

“Leo Baekeland sold the rights to his invention, Velox photographic printing paper, to Eastman Kodak in 1899. It was the first commercially successful photographic paper and he sold it to Eastman Kodak for \$1 million. Baekeland had planned to ask \$50,000 and to go down to \$25,000 if necessary, but fortunately for him, Eastman spoke first.”

1.1 Background and Motivation

Production scheduling is the backbone of many manufacturing information systems. The way a scheduling system is structured affects directly how production functions are carried out in a manufacturing system. Most existing methods for production scheduling take one of the two extreme approaches: (1) centralized or hierarchical decision making based on monolithic models, or (2) localized but uncoordinated decision making. Monolithic models that assume single-agent point of view with a centralized formulation of the overall decision making problem are often found in academic literature especially in Operations Research, while localized methods can be frequently observed in industry. In this study, we analyze both practical and theoretical requirements for a new generation of scheduling methods that coordinates distributed (or

1.1. BACKGROUND AND MOTIVATION

decentralized) locally autonomous decision making agents.

Due to the advancement of computing and information technology, decision making in modern manufacturing environment is becoming increasingly distributed. This is mainly due to two reasons: (1) local decision makers have direct access to decision making tools and computers, and (2) problems of primary interest involve multiple decision makers who are normally self-interested agents. Each decision maker is responsible for his/her own segment of production, i.e., each has his/her own constraints, special needs, preferences, and objectives, with some loose link to the global objective. Unavoidably, local decision makers may have conflicting interests, especially over their shared resources. For example, firms often have designated product managers each responsible for a set of similar product types. This product-oriented distribution of responsibilities and decision making are common in manufacturing systems with complex supply structures, such as the case in electronics and automotive manufacturing. Each product manager is motivated to satisfy his/her own customers' requirements. However, different market conditions, lead time pressure, and process requirements demand product managers to deal with distinctively different issues. In these environments, scheduling systems must accommodate local preferences and needs in order to support such objectives as market responsiveness, customer satisfaction, and pipe-line efficiency in the supply chain. Similar observations can be made for inter-firm supply chain management and synchronization efforts, maybe even in a more visible manner. With the increasing availability of web-based interface, electronic commerce, and collaboration tools, the problems of multiple decision makers with diverse interests seem to be more attractive for both practical and theoretical reasons.

The conventional approach to such “optimization” or “resource allocation” problems, at least in classical OR literature, has been to construct a *monolithic* model of the overall problem, solve it, and report the final solution as a “proposed action plan” for the problem as if there is a single agent who is capable of implementing the plan. Although the disadvantages of centralized or monolithic approaches in terms of impracticality and computational burden are somewhat well known, we have also been ignoring the two other important issues that exist in reality but are hard to

1.1. BACKGROUND AND MOTIVATION

capture with *conventional* approaches: (1) *decentralization of authority*, and (2) *dispersion of information*. The former refers to the multiplicity of the decision makers, i.e., there are multiple “agents” who are responsible from different segments of the decision making problem. The latter refers to the existence of the private information, i.e., the agents possess information that may be impractical to communicate with all details to a single decision maker or may be providing the agents a competitive advantage in which keeping or distorting the information serves for the agents’ local interests. From this point of view, the main problem becomes more of a *coordination* and *communication* problem rather than a single-point optimization problem. However, this does not mean that we must totally give up the optimization view, since it provides a system-wide metric that is a “bottom-line” or a “global performance measure” of the overall system. Thus, we need to address the following issues to be able to make a sound contribution in this field:

- In order not to lose the optimization view, i.e., to guarantee a high quality overall solution, we have to align the local interests with the system-wide objectives.
- Because of shared resources and private information, it is inevitable that the agents have conflicting interests. Hence, an important issue is to resolve peer-to-peer conflicts among agents.
- A single-point optimization view may impose solutions that are not equitable in any measure for any agent(s). This implies that the state implied by the solution may not be stable or in equilibrium. In this case, it is necessary to devise compensation or transfer mechanisms so that some sort of “stability” in the solution is achieved.
- When we follow a distributed approach for the problem, we need to address the communication and feasibility issues, which are automatically satisfied by a monolithic optimization model if used. Both the excess amount of communication and the interference by an outside authority (or a center) for feasibility restoration must be carefully analyzed.

1.2. DISSERTATION OUTLINE

- If the agents find beneficial for them to keep or distort their private information, and if our mechanism relies on the information that they report, then we need to make sure that we provide the agents the incentives that have them communicate their private information truthfully.

In this study, we attempt to address these issues as much as possible, in the context of *distributed scheduling* (It seems natural to see the examples of distributed decision making structure in most of the modern organizations. We will however work on a specific practical example on the following chapters, especially in the last two chapters, which is extracted from the scheduling environment of an automotive electronic component plant). With this global picture in mind, we organized the dissertation as a collection of four essays on distributed scheduling. We summarize the outline of the dissertation in the following section.

1.2 Dissertation Outline

We have organized the dissertation as a collection of interrelated papers. In each paper, we try to address a different facet of the complex distributed scheduling problem.

In Chapter 2, we approach the distributed scheduling problem from a mechanism (or algorithm) design angle and propose a *combinatorial auction* for the problem. The basic idea is to localize and distribute the functionality of operational scheduling decisions, leaving the complexity to local decision makers, while maintaining a simple and generic coordination mechanism at a central site. The proposed method is distributed in the sense that each local decision maker bases his/her decisions on a local utility which is defined based on both local preferences and global constraints. Specifically, each local decision maker has a local optimization problem which is to maximize expected total reward subject to his/her local constraints. This is then communicated to the coordination mechanism as a “bid”. The coordination mechanism or an “auctioneer” is a bid processor that makes resource allocation based on an iterative auction process using the bidding information. The auctioneer optimizes a global utility function which maximizes the efficiency of the overall system.

1.2. DISSERTATION OUTLINE

In this chapter, we also provide a rather extensive review of the relevant studies, which also serves as literature review chapter for the whole dissertation. We investigate our new method based on a distributed and locally autonomous decision structure using the notion of *combinatorial auction*. In *combinatorial auction* the bidders demand a combination of dependent objects with a single bid. We show that not only can we use this auction mechanism to handle complex resource scheduling problems, but there exist strong links between combinatorial auction and Lagrangean-based decomposition. Exploring some of these properties, we characterize combinatorial auction using auction protocols and payment functions. This study was our first step toward developing a distributed scheduling framework that maintains system-wide performance while accommodating local preferences and objectives. We provide some insights to this framework by demonstrating four different versions of the auction mechanism using job shop scheduling problems (Note that the agent and decision making structure developed in this chapter will be used in later chapters. This structure assumes that each decision agent is responsible from a job. We did not consider any machine agents since the job shop scheduling objective and constraints are job oriented and it is hard, if not impossible, to attach an economic utility and constraints for machine agents in the outlined structure. However, for a different setting of the scheduling problem, the agent design may involve only job agents, machine agents, or both).

In Chapter 3, we tested the robustness of the distributed approach that we proposed in Chapter 2. Thus, we study methods that improve scheduling robustness under uncertain disturbances and dynamic shop conditions. A new method is proposed based on the notion of preprocess-first-schedule-later [96]. Preprocessing starts at the beginning of the planning period where we use a Lagrangean relaxation of the scheduling model to form network-structured job subproblems. For each job subproblem we introduce stochastic constraints which capture *a priori* information in the form of processing time uncertainty. This stochastic information comes at minimal computational expense since the subproblems retain their special network structure. Using a subgradient search algorithm, we iteratively improve the lower and upper bounds of the scheduling instance. The preprocessing produces a partially

1.2. DISSERTATION OUTLINE

resolved sequence which we call Lagrangean ranking. Actual schedule generation is performed dynamically over time using the Lagrangean ranking established during preprocessing. Extensive computational experiments show that the proposed scheme significantly outperforms its deterministic counterparts without extra computational burden, improves the quality of dynamic dispatching, and maintains a robust performance under uncertain conditions.

In Chapter 4, we study the theoretical requirements for distributed scheduling problems. We consider a distributed decision process where interdependent decision agents compete for resources while making decisions based on their local utilities, preferences and constraints. We consider the case where each decision agent represents the interests of a job that has multiple operations to be processed in a job shop. We assume a price-directive mechanism for coordinating the job agents as we outlined in Chapter 2. In Chapter 4, we provide a rather detailed analysis of issues that we have uncovered in earlier chapters. These include *solution feasibility*, *information availability* and *communication overhead*. To address these issues, we introduce the notion of *proactive equilibrium*. The main idea is to make each agent consider an aggregate model of other agents in its utility maximization problem. Proactive equilibrium provides a basis that connects the theoretical underpinning between optimization and multiple agent models. We heavily make use of the structure that is special to the scheduling problem, and illustrate the details of the approach with an example constructed from real life scenarios

In any multi-agent (or distributed) environment, the involvement of the agents in the decision making process is crucial. If agents are not provided with proper incentives, they may not perform in a way that is aligned with the overall system goals. In Chapter 5, we analyze this important dimension of distributed decision making known as *incentive compatibility*. In spite of the apparent importance of this topic, it has been overlooked in most of the decentralized approaches to optimization. It has been generally assumed that agents always have the good will to cooperate and coordinate their decisions, and they provide all private information truthfully. Incentive compatibility deals with these issues explicitly in the design stage of the decision mechanisms. In this study, we concentrate on ways to aligning local interests with

1.2. DISSERTATION OUTLINE

global objectives and have them communicate their private information truthfully. To address the incentive compatibility issues we consider a *schedule selection game*. The agents possess private information about the true benefits of the schedules to their individual interests. We propose a direct revelation mechanism that implements an optimal schedule under dominant strategies. We then analyze the properties of the mechanism that are specific to our schedule selection game. We illustrate the details of the mechanism by a industry case study.

Chapter 2

Combinatorial Auction for Scheduling

2.1 Introduction

Distributed decision making has attracted much attention in recent years due to the rapid advancement of computing and communication technologies. Local decision makers responsible for a segment of the decision process now have direct access to powerful decision tools and workstations. The idea of making decisions in a heterogeneous, distributed fashion becomes both desirable and practical. Under such systems, decision makers may contemplate and incorporate their local set of constraints, preferences, and objectives into the decision process. Nonetheless, local considerations are loosely linked to global company interests, and decision makers often have conflicting interests over their shared resources. Two important research issues are thus, aligning global and local interests, and resolving peer-to-peer conflicts in resource sharing.

In this chapter, we will focus our attention on resource scheduling problems. Resource scheduling is a critical component of manufacturing information systems. The structure of a scheduling system affects directly how production functions are planned and carried out. Most existing methods for resource scheduling take one of the following two approaches: (1) centralized or hierarchical decision making based on some

2.1. INTRODUCTION

monolithic optimization model, or (2) distributed but ad hoc decision making designed to fit the existing organizational structure. Monolithic models are often found in the academic literature especially in Operations Research, while distributed or localized methods can be frequently observed in industry. We are interested in mechanisms that allow resource scheduling to be locally autonomous, and at the same time aligned with global interests.

The need for distributed resource scheduling is most evident in manufacturing systems with complex product/supply structures, and where customer service is of high priority. Electronics and automotive manufacturing fall into this category. In these operations, firms often have designated product managers each responsible for a set of similar product types. By design, each product manager is motivated to satisfy his/her own customers' requirements. Each product manager must deal with distinctively different sets of issues due to different market conditions, lead time pressure, and processing requirements. It is seldom economical to construct dedicated production lines for each product family using duplicated resources. Consequently product managers must share and compete for production resources on a regular basis. To satisfy the conflicting objectives of overall resource efficiency and individual product performance (e.g., market responsiveness, customer satisfaction), a new generation of scheduling system is desperately needed.

In this chapter, we study a distributed scheduling framework, which we believe is more suitable for the manufacturing decision structure. The basic idea is to localize and distribute the functionality of operational scheduling decisions, leaving the complexity to local decision makers, while maintaining a simple and generic coordination mechanism at a central site. The proposed method is distributed in the sense that each local decision maker bases his/her 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 his/her expected total reward subject to local constraints. This is then communicated to the coordination mechanism as a "bid". The coordination mechanism or an "auctioneer" is a bid processor that makes resource allocation based on an iterative auction process using the bidding information. We will demonstrate that traditional optimization models for scheduling can be viewed as special

2.2. A SURVEY OF RELATED LITERATURE

cases of the proposed auction structure, and well-known optimization techniques such as Lagrangean relaxation can be adapted to handle iterative auction.

2.2 A Survey of Related Literature

The notion of decentralized decision making can be found in early OR literature as well as in much recent work in economics and computer science. To put our proposed method in the context of existing literature, we will provide a survey of the latter two areas: agent-based scheduling which emerged from the computer science community, and auction theory and equilibrium which emerged from economics and game theory.

2.2.1 Distributed and Agent-based Scheduling

Distributed scheduling has attracted much attention recently in the computer science community. Methods based on *multi-agent systems* have emerged in the area of Distributed Artificial Intelligence (DAI). The idea is to decouple the scheduling task into subproblems each solved by a local agent. These approaches are rooted from the constraint satisfaction version of the scheduling problem [87, 52, 53]. The optimization version of the problem has rarely been attempted in DAI [87, 54]. Sycara, et al. [87] revise the micro-opportunistic scheduling approach studied in [78] to develop a distributed and multi-agent scheduling system. Liu and Sycara [54] propose a multi-agent approach for specially structured bottleneck job shop scheduling problems with a weighted tardiness objective.

Another line of research, also rooted from DAI, is *market-based* distributed problem solving. In this approach the communication among the problem-solving agents is governed by a computational economic process. The *contract net* approach proposed by Davis and Smith [22] is among the pioneer work in this area. The contract net model has been proposed for *dynamic on-line* manufacturing scheduling where an agent responsible for a manufacturing step issues a request-for-bids to all agents who can supply the input. The best bidder is then selected based on a certain criteria [70, 71, 1]. Upton, Barash and Matheson [89] study a market mechanism for

2.2. A SURVEY OF RELATED LITERATURE

parallel machine scheduling. They propose a particular bid calculation based on the earliest expected completion time. A similar auction-based approach for setup time determination is proposed by Wang and Veeramani [92].

Neiman, et al. [67] investigate a multi-agent distributed scheduling system where each agent is responsible for a set of resources. To perform its assigned tasks, an agent negotiates with other agents for its required resources. Sandholm [80] proposes a similar dynamic scheduling approach for the vehicle routing problem.

Another line of research in DAI uses the notion of economic equilibrium [94] to model and analyze the interactions among distributed agents. Wellman [93] proposes such a approach which finds equilibrium prices for multicommodity flow problems. More recently, Ygge and Akkermans [100] propose a resource-oriented equilibrium-seeking approach for power load management problems (this is a quantity tatonnement approach for Walrasian or competitive equilibrium). In similar studies, reflections of price equilibrium on market-based multi-agent systems are analyzed (see, e.g., [82, 83]). In a recent study, Wellman, et al. [95] develop price equilibrium for single-resource preemptive scheduling problems.

2.2.2 Lagrangean Relaxation Techniques

We will demonstrate that the distributed auction mechanism is a generalization of Lagrangean-based decomposition techniques for traditional optimization problems. Lagrangean relaxation (LR) techniques commonly used in optimization can be adapted to iterative auction mechanisms. In this section, we review some of the important developments in using LR for scheduling. There are two main uses of Lagrangean techniques: (1) They provide lower bounds that can be used in a branch-and-bound approach for an integer programming problem. (2) They can be used as a basis for designing efficient heuristics for hard-to-solve problems. We will focus on the latter.

Although LR techniques have many other application areas such as non-linear optimization, they were proposed for integer programming problems in the early 70s [68, 23]. Two papers by Fisher [20, 21] provide earlier reviews of LR techniques for

2.2. A SURVEY OF RELATED LITERATURE

integer programming. Before implementation of LR for JSP, it is used for lot sizing and production planning problems (see [25, 34, 75] and [15] for recent implementations in these areas). The other application area for LR is the single machine scheduling problem mainly with tardiness objective [19, 56]. An application to the single machine weighted completion time problem with precedence constraints is proposed in [77 and 32]. Parallel machine and flow shop applications are very few (See [40] for parallel machine problem, and [78] for 2-machine flow shop problem). Although applications of the method for the relatively simple scheduling problems are numerous, the JSP has been rarely attempted with LR.

One of the first LR implementations in the JSP-like context was made by Fisher [18] who used LR to get iteratively tighter lower bounds in each node of a branch and bound search. A specific JSP application of LR is proposed by Hoitomt, Luh and Pattipati [31] for a quadratic weighted tardiness objective. In this approach, the job shop scheduling problem is formulated as a discrete time integer programming, and both operation precedence constraints and machine capacity constraints are relaxed using Lagrangean multipliers to obtain operation level subproblems. Since the operation level objectives do not take into account the operation interdependence, they report serious oscillation problems observed during the course of subgradient optimization. They later propose another LR technique with artificial operation objectives with operational due dates and weights derived from the job due dates and the job weights [10]. The limited experimental study for randomly generated job shop scheduling problems shows some promising results. LR with dualized machine capacity constraints is used in another study to provide a basis for real-time dispatching by Roundy, et al. [64]. They propose a two-level scheduling approach in which the first level tries to find the best Lagrangean multiplier values corresponding to the machine capacity constraints (*machine prices*), and the second level dispatches the jobs with a priority rule that uses the computed machine prices. In this approach, they tried to capture the global view of scheduling in the first level (planning level) by calculating machine prices using subgradient optimization. Detailed scheduling is left to the second level (dispatching) which handles shop dynamics such as new job arrivals, and processing times variations.

2.2. A SURVEY OF RELATED LITERATURE

2.2.3 Auction Mechanism Design and Auction Algorithms

McAfee and McMillan [60] define auction as a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from market participants. The literature on auction theory flourished after the seminal paper by Vickrey [91], in which he introduces the concept of *second-price* auctions. A bibliography by Stark and Rothkopf [86] lists nearly 500 papers over the next two decades after Vickrey's study. In another survey, Engelbrecht-Wiggans [24] reviews main theoretical results and research efforts on auctions and bidding. Smith [85] discusses auction research in experimental economics published between 1962 and 1990. Most studies in this area focus on single-object simple auctions [62, 25]. A more recent review of competitive bidding with a broader scope can be found in [75].

If we consider scheduling problems as assigning time slots on unit-capacity machines (objects) to a set of competing jobs (bidders), we can see that it is at least as complex as *multi-object auctions* which deal with independent-valued, indivisible objects. Hylland and Zeckhauser [40] consider the problem of allocating individuals to positions with limited capacities. They propose an auction-assignment mechanism where the individuals respond to the system with true valuations. Leonard [51] investigates the problem of eliciting true preferences for an assignment problem in a similar setting. He presents a generalization of Vickrey's second-price auction mechanism. Demange, Gale and Sotomayor [23] propose a *dynamic* or *progressive auction* for the multi-item case to achieve incentive-compatible minimum price equilibrium. Instead of a single-pass sealed bid auction as in Leonard [51], they present two progressive auction mechanisms, one exact and another approximate. Sankaran [84] proposes an improvement on the exact auction mechanism based on a labeling algorithm.

The equivalence of multi-object auctions and the classical assignment problem led Bertsekas to develop the so called *auction algorithm* [5]. This algorithm is similar to the approximate auction mechanism proposed by Demange, Gale and Sotomayor [23]. Bertsekas shows that the price vector obtained at the end of the auction is the approximate optimal dual vector of the primal assignment problem. This, in fact, is a restatement of the result of Demange, Gale and Sotomayor. In this context, we can say that finding the equilibrium prices in an optimal auction design problem is

2.2. A SURVEY OF RELATED LITERATURE

equivalent to finding the dual optimal solution for the assignment problem. Bertsekas later extends his auction algorithm to other network flow problems such as shortest path, transportation and minimum cost flow [6, 7, 8].

In general scheduling problems a job agent may demand a combination of time slots to process its operations. Consequently we must deal with auctions in which agents bid for multiple items that have inter-dependent valuations. This motivates the investigation of *combinatorial auction* in which the bidders demand a set or a combination of indivisible objects with a single bid. Banks, Ledyard and Porter [2] consider allocating multiple resources each with *divisible* capacity when there are uncertainties in supply or demand. They generalize Vickrey's second-price auction for this environment where each bidder demands a combination of some portions of each resource with a single bid. Rothkopf, Pekec and Harstad [76] investigate simultaneous combinatorial auctions in which bidders submit bids for a combination of indivisible objects. They show that finding the revenue-maximizing set of nonconflicting bids is NP-Hard. They examine different bid structures to find computationally tractable auction types. Sandholm [81] also analyzed the winner determination problem with its practical implications on electronic commerce and Internet-based auctions. Applications of auction mechanisms to optimization has also been proposed by Graves, Schrage, and Sankaran [31] for class scheduling, and Rassenti, Smith, and Bulfin [74] for the allocation of airport time slots to airlines.

In a majority of these problems, competitive (Walrasian) price equilibria are known to exist and auction procedures aim to reach one of these equilibria. In this respect, price-directed auctions can be viewed as tâtonnements designed to reach market clearing (equilibrium) prices in an exchange economy [59]. Several researchers have studied the correspondence between equilibrium conditions and the convergence properties of auction. Most of this work was done in the context of an exchange economy with interdependent and indivisible objects (c.f., [9, 55, 56, 38, 39]). These studies show that a price equilibrium may not exist in general combinatorial auctions where agents demand a bundle of interdependent items. Gul and Stacchetti [39] show that in such an economy unless the agents' utility functions satisfy conditions called *gross substitutes* or *no complementarities* the existence of a price equilibrium is not

guaranteed. Bikhchandani and Mamer [9] show that market clearing prices exist *if and only if* the solution value of a centralized integer program coincides with that of its LP relaxation. This condition significantly lowers the expectation of reaching price equilibrium using any simple auction mechanism.

2.3 Problem Statement and Solution Methodology

2.3.1 Problem Statement

We consider the classical job shop scheduling problem (JSP) where a set of jobs is to be completed and each job requires a set of machines for a certain period of time for processing. Each *job* consists of a series of operations that represent the production steps of the job. Each *operation* needs a certain *machine* for a certain time period called *processing time*. Once an operation is started, no interruptions are allowed until completion, i.e., no preemption. Job processing times and their corresponding machine requirements represent the job *routing*. The prescribed sequence of operations in each job defines the *precedence constraints*. Each machine has a capacity of one, i.e., it can process one and only one operation at a time. The scheduling objective is to minimize total weighted tardiness. When all jobs have identical routing, JSP is reduced to a flow shop scheduling problem. If each job requires only a single stage operation on one particular machine, or a dominant resource forms the bottleneck, then JSP is reduced to a single machine problem. For a complete taxonomy of production scheduling problems, see Pinedo [72].

2.3.2 A Combinatorial Auction Mechanism for Scheduling

We propose a general auction mechanism for JSP using the notion of multi-item combinatorial auction. The proposed mechanism is progressive or dynamic [23] since it involves some number of iterations before allocating objects to bidders. We define “objects” as discrete time slots on the machines, and “bidders” as individual jobs. An

Table 2.1: Notation

i	Job (bidder) index, $i = 1, \dots, N$ where N is number of jobs
j	Operation index, $j = 1, \dots, n_i$ where n_i is number of operations of job i
t	Time slot index, $t = 1, \dots, T$ where T represents the length of the planning horizon during which all the jobs can be completed (i.e. some reasonable upper bound on the makespan of the problem).
k	Machine index, $k = 1, \dots, M$ where M is number of machines (Hence there are TM time slots for bidding).
W_i	Weight or tardiness penalty of job i
d_i	Due date of job i
o_{ik}	The operation of job i which requires machine k
m_{ij}	Machine required for operation j of job i , i.e. $o_{ik} = j$ if $m_{ij} = k$
p_{ij}	Processing time of operation j of job i
$B_{ij;a,b}$	Operation bid, i.e. a combination of time slots from time slot a to time slot b for operation j of job i
B_i	Job bid, i.e. collection of operation bids (A combination of time slots demanded by job i)

auctioneer is the coordinating agent or the seller who iteratively updates the prices of objects starting from initial reservation prices. Based on current pricing, each job tries to find the best combination of time slots on the machines (objects) so as to maximize its own utility function. The auctioneer evaluates bids from all jobs and updates the reservation prices according to the conflicts among their requests. This process repeats in an iterative fashion until it finds a conflict-free allocation [23]. Obviously, objects have inter-dependent values, and different combination of objects presents different values for the jobs as is the case in combinatorial auctions [74, 76]. The precedence and non-preemption constraints restrict the combination of time slots that each job can bid.

In the following, we state more formally the price directed auction procedure for the weighted tardiness job shop scheduling problem. We adopt the notation shown in Table 2.1. The set of objects (time slots available from the machines) can be defined as a set of pairs (machine, time slot). Hence each possible bid B_i from job i is a

subset of the following object set:

$$O = \{(k, t) : 1 \leq k \leq M, 1 \leq t \leq T\} \quad (2.1)$$

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

$$O_{m_{ij}} = \{(m_{ij}, t) : 1 \leq t \leq T\} \quad (2.2)$$

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

$$B_{ij;a,b} = \{(m_{ij}, t) : 1 \leq a \leq t \leq b \leq T, b = a + p_{ij} - 1\} \quad (2.3)$$

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

$$B_i = \bigcup_{j: a_{i,j+1} > b_{i,j}} B_{ij;a_{ij},b_{ij}} \quad (2.4)$$

The condition in the set definition determines the precedence constraints between consecutive operations. We will denote the set of all allowed locally feasible bids (i.e., those satisfying non-preemption and precedence constraints) as $\mathcal{B}_i$.

A job incurs a tardiness cost if its completion is past due, and the job pays a fee for using the resources at its demanded time slots. We thus define job i 's utility function as follows:

$$U_i(B_i) = -W_i \max\{0, b_{i,n_i} - d_i\} - P_i(B_i) \quad (2.5)$$

where $P_i(B_i)$ is the total payment if the demanded time slots in bid B_i are allocated to job i . The first term in the utility function accounts for the total weighted tardiness attributed to job i by demanding B_i which in turn yields the completion of the job at b_{i,n_i} . The total payment P_i is a function of the time slots demanded in bid B_i . Hence, each job agent must trade off possible savings on due-date performance with payments due to resource usage. The best bid for job i (B_i^*) is one that maximizes

the utility function, i.e.,

$$\max_{B_i \in \mathcal{B}_i} U_i(B_i). \quad (2.6)$$

The center has several options in regard to the payment function P_i . A most straight forward payment scheme is *regular tâtonnement* where all job agents are subject to the same linear payment function. There is no price discrimination among job agents and they are charged the same price for a particular (machine, time slot) pair. However, as will become apparent in the following section, under *regular tâtonnement* there is no “optimal pricing” that supports optimal resource allocation. As demonstrated in the context of large-scale linear programming decomposition [42, 43], linear subproblem objective leads to degeneracy. To relieve the problems of degeneracy, Jose, Harker and Ungar [43] propose *augmented tâtonnement* which imposes price discrimination among job agents using a quadratic perturbation term [42] in the subproblem objective. As a result, the unit price paid by a specific job agent varies depending on the particular combination of resource usage it demands. Based on this distinction in the payment (P_i), in turn in the utility function, we define two types of tâtonnement: *regular*, and *augmented tâtonnement*.

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^*). All 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 $B_i^* \equiv B_i^*(\lambda)$ where $\lambda = (\lambda_1, \dots, \lambda_M)$, computed from current machine-time slot prices $(\lambda_{kt}, k = 1, \dots, M, t = 1, \dots, T)$ announced by the auctioneer.

The goal of price updates is to reduce *resource conflicts* among job agents. A conflict exists if one time slot (object) is demanded by more than one job. Since each bid B_i defines a demand point for job i , one way of updating prices in this procedure is to adjust them according to the *excess job demand* (number of jobs that bid for a certain time slot minus the total capacity of the machine). One possible strategy is

to have the auctioneer raise the prices in proportion to the excess demand as follows:

$$D_{kt} = \sum_{i=1}^N \delta_{ikt}^* - 1 \quad (2.7)$$

where δ_{ikt}^* is 1 if job i demands time slot (k, t) in its optimal bid (i.e. if $(k, t) \in B_{ij; a_{ij}, b_{ij}}^*$ and $m_{ij} = k$), and is 0 otherwise. Since excess demand can be negative, it is possible to reduce prices for time slots which do not have enough demand. Since no object can be sold with a negative price, we consider only nonnegative prices. Thus we define the price adjustment as follows:

$$\lambda_{kt}^{r+1} = \max \{0, \lambda_{kt}^r + f(D_{kt}^r)\} \quad (2.8)$$

where r is the iteration number, and f is the *price adjustment function* nondecreasing in current excess demand D_{kt}^r . Since a large variety of functions satisfy this property, this form of price adjustment is quite flexible. Based on the form of f we can define different *auction protocols* that govern the progress from one iteration to the next.

One such auction protocol is called *standard Walrasian tâtonnement* in which the adjustment function is defined as a constant multiplier times the current excess demand:

$$f(D_{kt}^r) = s D_{kt}^r \quad (2.9)$$

where s is called *price adjustment factor* or *step parameter*. This form of tâtonnement with a carefully selected step is known to converge to an optimal (equilibrium) allocation in a pure exchange economy. However, when the demand is discontinuous with interdependencies, one can no longer expect the convergence property to hold in general. Since the step parameter is constant over iterations we call this a *non-adaptive tâtonnement*. The term is defined in contrast to the *adaptive tâtonnement* where the auctioneer makes aggressive price update in early iterations of the auction to quickly assess the overall demand status among job agents. This is followed by smaller adjustments (low values of s) in later iterations to fine tune the quality of allocation. In this case, the amount of price adjustment changes depending on the current iteration and the progress of the auction.

With the above discussion, we characterize the combinatorial auction by the following elements (for more formal definitions of the terms, see Section 2.3.5):

1. Auction Protocols

- *Standard Walrasian tâtonnement* (Non-Adaptive)
- *Adaptive tâtonnement*

2. Payment Functions

- *Regular tâtonnement* (no price discrimination)
- *Augmented tâtonnement* (with price discrimination)

In the next section, we briefly describe a game-theoretic model of this auction-based allocation scheme.

2.3.3 A Game Theoretic Perspective

We provide a game theoretic interpretation of the above auction model based on the relationship between competitive (or Walrasian) economy and game theory. The model uses the notion of *noncooperative pseudogame* or *abstract economy* proposed by Freedman [28] and Ichiishi [41]. In this game, a player i is influenced by other players in two ways: (1) indirect influence, where i 's feasible strategies is restricted to a subset, and (2) direct influence, where i 's utility level is affected. Consider $N + 1$ players, where players 1 to N are the job agents, and player 0 is the auctioneer. We assume that each job agent has enough endowment to buy any collection of time slots (i.e., no budget restrictions). Job agent i 's consumption set (allowed bids) is also its strategy set, i.e., $S_i = \mathcal{B}_i$. The auctioneer chooses a nonnegative price vector from a strategy set defined as

$$S_0 = \{\lambda_{kt} : 0 \leq \lambda_{kt} \leq \Lambda, k = 1, \dots, M, t = 1, \dots, T\},$$

where Λ is an upper bound value for prices, e.g., a value that is known to derive the total demand to zero. Given a pair (B, λ) , we define the restricted (or feasible)

strategy set of player i ($i = 1, \dots, N$) as

$$F_i(B, \lambda) = \{B'_i : \delta'_{ikt} \leq 1 - \sum_{i' \neq i} \delta'_{i'kt}, \forall k, t\}$$

where δ'_i is defined as before according to B'_i . The feasible strategy set of player 0 is not affected by other players, i.e. $F_0 = S_0$. The utility function of player i is defined as

$$u_i((B, \lambda), B'_i) = U_i(B'_i), \forall B'_i \in F_i(B, \lambda).$$

Here, $u_i((B, \lambda), B'_i)$ is agent i 's utility when it chooses B'_i and all others choose

$$(B_1, \dots, B_{i-1}, B_{i+1}, \dots, B_N, \lambda).$$

The utility function of player 0 is

$$u_n((B, \lambda), \lambda') = \sum_k \sum_t \lambda'_{kt} (\sum_i \delta_{ikt} - 1)$$

One interpretation of this utility maximization is that the auctioneer seeks to minimize the total value of the excess capacity of the machines: $\sum_k \sum_t \lambda_{kt} (-D_{kt})$. Then, the triplet $(\{S_i\}_{i=0}^N, \{F_i\}_{i=0}^N, \{u_i\}_{i=0}^N)$ defines a pseudogame.

It is easy to show that the game corresponding to a classical exchange economy has a strategy equilibrium and that this is a competitive equilibrium. However, this is only true when the utility functions are concave and the strategy sets are convex. Unfortunately these conditions are rarely true in general scheduling problems (see [95] for the existence of equilibrium pricing in one-machine scheduling problems). Nonetheless, the game theoretic perspective introduces potentially interesting issues for distributed scheduling.

2.3.4 Lagrangean Relaxation

We will now turn our attention to an integer programming (IP) formulation of the job shop scheduling problem with weighted tardiness objective. Although there are several IP formulations for JSP we will use the discrete time formulation due to

Pritsker, Watters, and Wolfe [73]. The decision variable is X_{ijt} , where

$$X_{ijt} = \begin{cases} 1 & \text{if operation } j \text{ of job } i \text{ has started by time } t, \\ 0 & \text{otherwise.} \end{cases} \quad (2.10)$$

The JSP under consideration can be formulated as follows:

(JSP)

$$\min \sum_i W_i \left[\sum_{t > d_i - p_{i,n_i}} (1 - X_{i,n_i,t}) \right] \quad (2.11)$$

$$\text{s.t. } X_{i,j,t+1} \geq X_{i,j,t}, \quad \forall i, j, t < T \quad (2.12)$$

$$X_{i,j,t} \leq X_{i,j-1,t-p_{i,j-1}}, \quad \forall i, j > 1, t \quad (2.13)$$

$$\sum_{i,j:m_{ij}=k} (X_{i,j,t} - X_{i,j,t-p_{ij}}) \leq 1, \quad \forall k, t \quad (2.14)$$

$$X_{i,j,t} \in \{0, 1\}, \quad \forall i, j, t \quad (2.15)$$

The first set of constraints (2.12) makes sure that once an operation is started, it remains so in all subsequent time periods. This is due to the definition of X_{ijt} as outlined in (2.10) and the non-preemption requirement. Constraints (2.13) state that an operation cannot start until all its predecessors are completed. Finally, the machine capacity constraints (2.14) state that at most one job can be processed on a particular machine in a given time period. The objective function (2.11) is the total weighted tardiness derived from the fact that a cost of W_i is incurred for each time period after d_i during which job i has not been completed.

The objective function can be rewritten as follows:

$$\sum_i W_i \left[\sum_{t > d_i - p_{i,n_i}} (1 - X_{i,n_i,t}) \right] = \sum_i W_i (T - d_i + p_{i,n_i}) + \sum_i \sum_j \sum_t -w_{ijt} X_{ijt} \quad (2.16)$$

where

$$w_{ijt} = \begin{cases} W_i & \text{if } j = n_i \text{ and } t > d_i, \\ 0 & \text{otherwise.} \end{cases} \quad (2.17)$$

Since the first term in the objective function (2.16) is a constant we represent it as

$A = \sum_i A_i$ where $A_i = W_i(T - d_i + p_{i,n_i})$.

When we dualize the machine capacity constraints (2.14) of (JSP) with nonnegative Lagrangean multipliers $\lambda_{kt}, \forall k, t$ we obtain the following Lagrangean Relaxation problem:

(LR_λ)

$$\min \sum_i A_i + \sum_i \sum_j \sum_t (\lambda_{m_{ij},t} - \lambda_{m_{ij},t+p_{ij}} - w_{ijt}) X_{ijt} - \sum_k \sum_t \lambda_{kt} \quad (2.18)$$

subject to (2.12), (2.13), and (2.15)

We decompose LR_λ into independent job-level subproblems as follows:

$$v(LR_\lambda) = \sum_i v(LR_{\lambda,i}) - \sum_k \sum_t \lambda_{kt} \quad (2.19)$$

where $v(P)$ denotes the value of optimal solution of problem P , and

$$v(LR_{\lambda,i}) = \min A_i + \sum_j \sum_t (\lambda_{m_{ij},t} - \lambda_{m_{ij},t+p_{ij}} - w_{ijt}) X_{ijt} \quad (2.20)$$

$$\text{s.t. } X_{i,j,t+1} \geq X_{i,j,t} \quad \forall j, t < T \quad (2.21)$$

$$X_{i,j,t} \leq X_{i,j-1,t-p_{i,j-1}} \quad \forall j > 1, t \quad (2.22)$$

$$X_{i,j,t} \in \{0, 1\} \quad \forall j, t \quad (2.23)$$

Each subproblem satisfies the integrality property, in fact each is the dual of a specially structured maximum flow network problem [77]. Hence, each job-level subproblem can be solved using efficient network flow algorithms.

For a vector of Lagrangean multipliers λ , $v(LR_\lambda)$ which is a function of solutions to $(LR_{\lambda,i})$, provides a lower bound on the optimal cost of (JSP) . The best lower bound corresponds to the solution of the following Lagrangean dual problem:

$$(LRD) \quad \max_{\lambda \geq 0} v(LR_\lambda)$$

There are alternative ways to improve the Lagrangean lower bound and solve the Lagrangean dual. One method is the subgradient search procedure [69]. Subgradient optimization starts with an initial value for the multipliers λ , e.g. $\lambda^0 = 0$. Then the method generates a sequence λ^r over the iterations r by the rule

$$\lambda_{kt}^{r+1} = \max\{0, \lambda_{kt}^r + s_r \gamma_{kt}(X^{*r})\} \quad (2.24)$$

where s_r is an appropriately selected step size and $\gamma_{kt}(X^{*r})$ is the subgradient of the capacity constraint of machine k at time period t defined by the optimal solution X^{*r} of LR_{λ^r} :

$$\gamma_{kt}(X^{*r}) = \sum_{i,j:m_{ij}=k} (X_{i,j,t}^{*r} - X_{i,j,t-p_{ij}}^{*r}) - 1 \quad (2.25)$$

Step size s_r should satisfy certain conditions in order to get a good convergence to optimal Lagrangean dual value. A formula that has proven effective in practice is as follows [26]:

$$s_r = \alpha_r \left(\frac{UB - v(LR_{\lambda^r})}{\sum_k \sum_t \gamma_{kt}^2(X^r)} \right) \quad (2.26)$$

where α_r is a scalar satisfying $0 < \alpha_r \leq 2$ and UB is a target upper bound value for Lagrangean dual which can be updated over the iterations.

We may update the upper bound value using a fast heuristic that converts the capacity-infeasible solution (due to resource conflicts) to a feasible one in each iteration. It is also suggested in the literature that one should decrease the value of α if there is no improvement over a certain number of iterations [26]. The procedure stops when (1) a capacity-feasible global solution is found for problem LR_{λ^r} , (2) a lower bound is epsilon-close to the upper bound, (3) α becomes very small, or (4) a specified number of iterations is reached. Typically the procedure ends with an infeasible solution due to the last three criteria.

2.3.5 Combinatorial Auction and Lagrangean Relaxation

In this section, we show that subgradient search in the context of Lagrangean Relaxation can be viewed as a particular version of combinatorial auction introduced

in Section 2.3.2. We focus our attention on the payment functions and the auction protocols.

Payment Functions

We first establish a basic property connecting Lagrangean relaxation with the *regular tâtonnement* version of combinatorial auction. Using the well-known properties of Lagrangean duality we can then explore alternative payment functions.

Theorem 2.1 *For a given resource price vector λ , let the payment function P_i be the sum of the current resource prices for the slots included in a bid B_i (as in regular tâtonnement), i.e.,*

$$P_i(B_i) = \sum_{j=1}^{n_i} \sum_{t=a_{ij}}^{b_{ij}} \lambda_{m_{ij},t} \quad (2.27)$$

Then, the following solution $X_{ijt} \forall j, t$ defined using the optimal bid $B_i^*(\lambda)$ of job agent i solves the job-level network flow problem $LR_{\lambda,i}$ defined by (2.20)–(2.23).

$$X_{ijt} = \begin{cases} 1 & \text{if } t \geq a_{ij}^* \\ 0 & \text{otherwise.} \end{cases} \quad (2.28)$$

Proof. First, note that the constraints (2.21)–(2.23) are the same constraints that define the set $\mathcal{B}_i$ of allowed bids B_i . If we substitute the defined payment function (2.27) into the utility function $U_i(B_i)$ (see Equation (2.5)), we obtain

$$U_i(B_i) = -W_i \max \{0, b_{i,n_i} - d_i\} - \sum_{j=1}^{n_i} \sum_{t=a_{ij}}^{b_{ij}} \lambda_{m_{ij},t}$$

We note that, for a given λ , this is the negative of the objective (2.20) of the job-level problem ($LR_{\lambda,i}$). Then, the maximum value of the constrained utility function is the same as the minimum value of the job-level objective in LR. Since $B_i^*(\lambda)$ is optimal, it solves the utility maximization problem defined by (2.6). Hence,

$$U_i(B_i^*) = -W_i \max \{0, b_{i,n_i}^* - d_i\} - \sum_{j=1}^{n_i} \sum_{t=a_{ij}^*}^{b_{ij}^*} \lambda_{m_{ij},t} = v(LR_{\lambda,i})$$

where a_{ij}^* and b_{ij}^* respectively represent the first and last time slots for each operation j included in the bid B_i^* . That is, a_{ij}^* defines the optimal starting time of the operation j of job i in optimal bid B_i^* . Hence, can construct the solution (2.28) using the definition of decision variables X .

Theorem 2.2 *Let λ^* be the optimal resource prices corresponding to the Lagrangean Dual Problem (LRD), i.e.,*

$$\lambda^* = \arg \max_{\lambda \geq 0} v(LR_\lambda),$$

and $B_i^*(\lambda^*)$ be the optimal bids for price vector λ^* where

$$U_i(B_i^*) = v(LR_{\lambda^*}).$$

Locally optimal bids $B_i^*(\lambda^*)$ solve (JSP) if and only if λ^* and B_i^* satisfy the Strong Lagrangean Duality Theorem [69].

Proof. This follows directly from Theorem 2.1 and the well-known Strong Lagrangean Duality Theorem.

Only in rare cases do discrete optimization problems satisfy the requirement of Strong Lagrangean Duality. In the case of (JSP), a significant duality gap between $v(LR_{\lambda^*})$ and $v(JSP)$ is to be expected. This result implies that there are unlikely to be optimal resource prices (therefore bids) that would support the optimal solution to JSP. In other words, if linear resource pricing with no price discrimination is used in combinatorial auction (as is the case in *regular tâtonnement*), the auction procedure does not lead to an optimal resource allocation. A market clearing mechanism (in this case, a feasibility restoration heuristic, or a branch and bound algorithm) will be necessary at the end of the auction to resolve the resource conflicts. As discussed in Section 2.3.2, even in the case of a large scale linear program (which does satisfy Strong Lagrangean Duality at optimality), there may not be resource prices supporting the optimal solution due to subproblem degeneracy.

The above results motivate the consideration of price discrimination in the job payment function, which leads to an *augmented tâtonnement* [42, 43]. In this case,

the center announces a *price schedule* instead of a price vector:

$$\lambda'_{kt} = \lambda_{kt} + q\delta_{ikt} \quad (2.29)$$

where λ_{kt} is the original price, q is a carefully selected small constant for better convergence, and δ_{ikt} is defined as before (i.e., 1 if time slot t on machine k is demanded by job i , 0 otherwise). Thus, job i 's payment function is defined as:

$$P_i(B_i) = \sum_{j=1}^{n_i} \sum_{t=a_{ij}}^{b_{ij}} \lambda'_{m_{ij},t} \quad (2.30)$$

This change is reflected in the objective of the i 'th subproblem in LR as follows:

$$A_i + \sum_j \sum_t -w_{ijt}X_{ijt} + \sum_j \sum_t \lambda_{m_{ij},t}(X_{ijt} - X_{i,j,t-p_{ij}}) + \sum_j \sum_t q(X_{ijt} - X_{i,j,t-p_{ij}})^2 \quad (2.31)$$

However, the standard price schedule (2.29) applied to (*JSP*) has an anomaly as the quadratic perturbation term added to the objective joins to the linear term (with the multiplier q) since

$$(X_{ijt} - X_{i,j,t-p_{ij}})^2 = X_{ijt} - X_{i,j,t-p_{ij}}.$$

To address this issue, we define a different perturbation scheme which adds a quadratic term for each *time zone* defined as a combination of a fixed number of consecutive time slots. The revised objective is as follows:

$$A_i + \sum_j \sum_t -w_{ijt}X_{ijt} + \sum_j \sum_t \lambda_{m_{ij},t}(X_{ijt} - X_{i,j,t-p_{ij}}) + \sum_j \sum_{h=1}^H q \left(\sum_{t=T_h}^{T_h+L-1} X_{ijt} - X_{i,j,t-p_{ij}} \right)^2 \quad (2.32)$$

where H is the number of time zones each with length L and beginning time of a zone h is defined by T_h . We can then define the time zone capacity as the sum of L consecutive capacity constraints in (2.14) (i.e., there can be at most L jobs in a time zone of L consecutive time periods):

$$\sum_{t=T_h}^{T_h+L-1} \sum_{i,j:m_{ij}=k} (X_{i,j,t} - X_{i,j,t-p_{ij}}) \leq L, \quad \forall k, t \quad (2.33)$$

We now have two alternative utility functions defined based on the specific payment function set by the center:

- *Regular tâtonnement* where payment function P_i is defined as in Equation (2.27).
- *Augmented tâtonnement* in which payment function P_i includes a quadratic perturbation defined in Equation (2.32).

Note that for the latter case each job agent needs to solve a quadratic subproblem. This implies a higher computational cost per iteration, but the overall convergence might compensate for this extra cost.

Auction Protocols

As summarized earlier, auction protocols can be implemented in a non-adaptive *standard Walrasian* fashion, or as an *adaptive tâtonnement*. The subgradient optimization (Section 2.3.4) can be viewed as a particular version of the latter. While the subgradient optimization tries to penalize infeasibility of dualized constraints, the auctioneer updates the resource prices so as to discourage conflicts on the demanded objects. While the subgradient algorithm adjusts the prices proportional to the amount of infeasibility, the auction algorithm updates the prices proportional to desirability of the object (measured in terms of excess demand). In fact, the subgradient (2.25) defined for the optimization procedure can be set equal to the excess demand (2.7) used in the auction:

$$\gamma_{kt}(X^{*r}) = D_{kt}^r. \quad (2.34)$$

Then, one adaptive way of updating prices is to set the adjustment function f in (2.8) as below:

$$f(D_{kt}^r) = s_r D_{kt}^r \quad (2.35)$$

In this case, the auction algorithm can be set equivalent to the subgradient optimization. Moreover, the basic principle of price adjustment in an *adaptive tâtonnement* is parallel to the idea of subgradient price adjustments. The proven convergence properties of subgradient search are particularly attractive. Nonetheless, at the end of the

2.4. EXAMPLES

subgradient optimization a significant duality gap is to be expected and a feasibility restoration heuristic must be used to eliminate the positive excess demand.

We now define in more specific terms the two auction protocols based on the definition of the price adjustment function (2.8):

- *Non-adaptive tâtonnement*, $f(D_{kt}^r) = sD_{kt}^r$.
- *Adaptive tâtonnement*, $f(D_{kt}^r) = s_r D_{kt}^r$ where s_r is the step size defined in (2.26).

The combinatorial auction mechanism for scheduling is summarized in Figure 2.1. As shown in the procedure, the auctioneer is not only responsible for price update in each iteration, it also performs feasibility restoration in each iteration. In this way, the auctioneer may announce the best potential feasible schedule in case the auction terminates early. This can be viewed as an upper bound calculation routine used in each iteration. Note that each job agent's local problem is a maximum flow problem, for which there exist polynomial algorithms.

2.4 Examples

Combining the two different payment functions and the two auction protocols leads to four different versions of combinatorial auction. In the following, we use small job shop examples to illustrate the implementation details of the proposed auction algorithm. We are also interested in gaining insights on the convergence behavior as well as the quality (duality gap) of the final solution when the auction terminates.

We first study small (3 machines, 3 jobs, or 3x3) scheduling problems. Since the routing structure of the jobs dictates both the level and the sequence in which the jobs demand for resources, we expect the auction mechanism to behave quite differently under different routing structures. We choose three different routing structures for our examples: (1) *random job shop*: the sequence each job visits the machines is random, (2) *bottleneck job shop*: the routing is random as in (1), but all jobs require longer processing times on a specific bottleneck machine, and (3) *flow shop*: the jobs

2.4. EXAMPLES

- Step 1. Initialization: The auctioneer initializes the machine-time slot prices λ , ($\lambda_{kt}^0 = 0, \forall k, t$); set iteration counter $r = 0$, scalar parameter α_r ($\alpha_0 = 2$), and the multiplier of the quadratic term ($q = 0.1$) (for augmented tâtonnement only).
- Step 2. Each job agent solves the job-level utility maximization subproblem (using either one of the payment functions) then submits the its optimal bid ($B_i^*(\lambda^r)$) to the auctioneer. Each bid corresponds to a job-level schedule.
- Step 3. The auctioneer combines all the bids and generates a capacity-infeasible shop-level schedule. The summation of total weighted tardiness and total potential payments from the jobs gives a lower bound (LB_r) for the original JSP.
- Step 4. The auctioneer converts this capacity-infeasible schedule into a feasible one by resolving the resource conflicts. Using this feasibility-restored shop schedule, the auctioneer computes corresponding total weighted tardiness as an upper bound (UB_r).
- Step 5. The auctioneer updates the best upper bound value found so far, $UB \leftarrow UB_r$, and records the best feasible shop schedule. The auctioneer updates α_r if necessary (Halve it if there is no improvement in LB_r in last 3 iterations).
- Step 6. The auctioneer computes the excess demand vector (D), (step size s_r in the case of *adaptive tâtonnement*) and updates time-slot prices λ^{r+1} using one of the protocols.
- Step 7. The auctioneer checks if a stopping criterion is satisfied. If not, the auctioneer starts the next iteration ($r \leftarrow r + 1$, go to Step 2). Otherwise, he stops, and announces the best feasible schedule found.

Figure 2.1: An iterative combinatorial auction mechanism for resource scheduling

2.4. EXAMPLES

Table 2.2: 3x3 Random job shop example

Job	Weight	Due date	Machine (Processing time)
1	4	10	1 (3), 2 (1), 3 (6)
2	6	10	3 (3), 1 (7), 2 (1)
3	2	12	1 (2), 3 (4), 2 (4)

visit the machines in the same order, but require different processing times. We investigate each of the three routing types in the following sections.

2.4.1 Random job shop

The data for this problem is shown in Table 2.2 (This is a modified version of the 3x3 job shop example in [50]). First, we implement the *adaptive regular tâtonnement* in which the payment function has no price discrimination and the auction protocol is adaptive as in subgradient search. We set the planning horizon (T) as 30 time periods. Hence, we have 30 time slots (objects) in each machine and 90 objects in total that the job agents must bid for. The capacity-infeasible shop schedule constructed by the auctioneer after the first iteration is shown in Figure 2.2. As shown in the figure, the job-level schedules satisfy non-preemption and precedence constraints, the machine capacity constraints are violated in a few time slots. The auctioneer will increase the prices of these time slots in the following iteration and thereby reducing the demand. The feasibility restored schedule after iteration 4 is shown in Figure 2.4.

For the examples problems in this section, we restricted the number of iterations (K) to 30. The price profile for each machine at the end of the auction is shown in Figure 2.3. As we can see, each machine has a considerably different price profile over the time slots. The adaptive subgradient price update essentially captures the desirability of each time slot on each machine. At the end of this adaptive tâtonnement, the auctioneer announce the best overall schedule generated over the iterations. Figure 2.5 plots LB and the best UB values versus iterations. Major improvements on LB are in early iterations. The final duality gap for this problem is 14.6%. For this example problem, the best UB schedule with the total weighted tardiness of 22 (achieved in

2.4. EXAMPLES

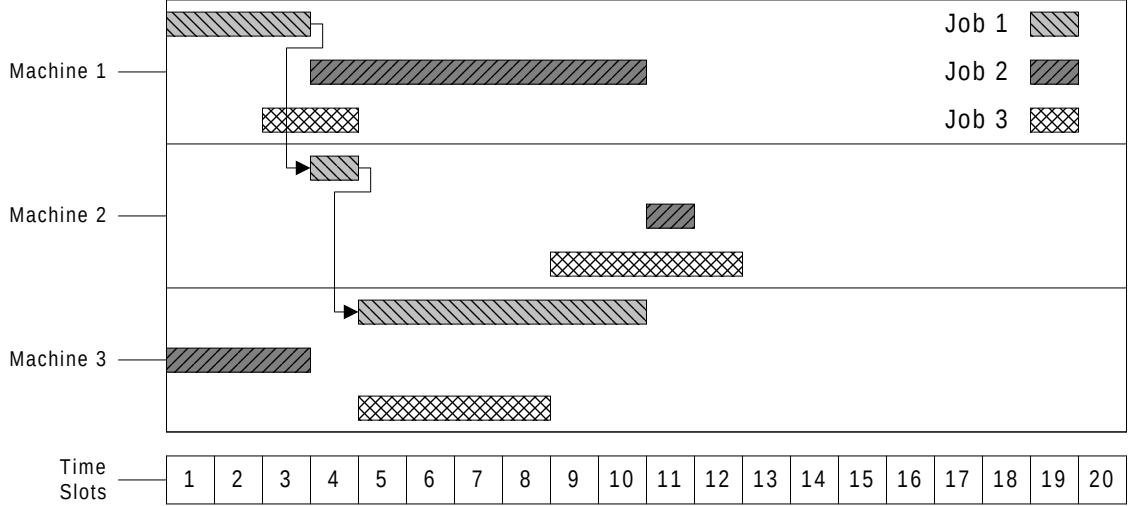


Figure 2.2: Infeasible shop schedule after combining the job bids (first iteration)

iteration 4) is optimal (The optimality proof is done outside the algorithm by solving the problem monolithically; it is shown in Figure 2.4).

The iterative process in *non-adaptive tâtonnement* is similar to the adaptive version except that the step parameter (s) is constant over the iterations. We experimented with several s values but we will only summarize the results for the case of $s = 0.2$. The price profile achieved at the end is similar to that shown in Figure 2.3. The change in LB and UB over iterations is plotted in Figure 2.6. While LB improves slower than in the adaptive process, the best UB is achieved in a later iteration (9th).

We next study *augmented tâtonnement* with non-adaptive and adaptive price updates. Recall that the augmented approach requires the use of “time zones.” We constructed each time zone by combining two consecutive time slots, i.e., $L = 2$ and $H = 15$. In *augmented tâtonnements*, the selection of the parameter q directly affects the convergence behavior. For the example problems, we experimented with several values and fixed q at 0.1. For the *adaptive augmented tâtonnement*, note that the LB and UB are now computed by solving job agent subproblems with a revised quadratic objective function. The revised LB and UB values are shown in Figure 2.7 with a final duality gap of 23.4%. The best restored feasible schedule (in fact, it is the optimal

2.4. EXAMPLES

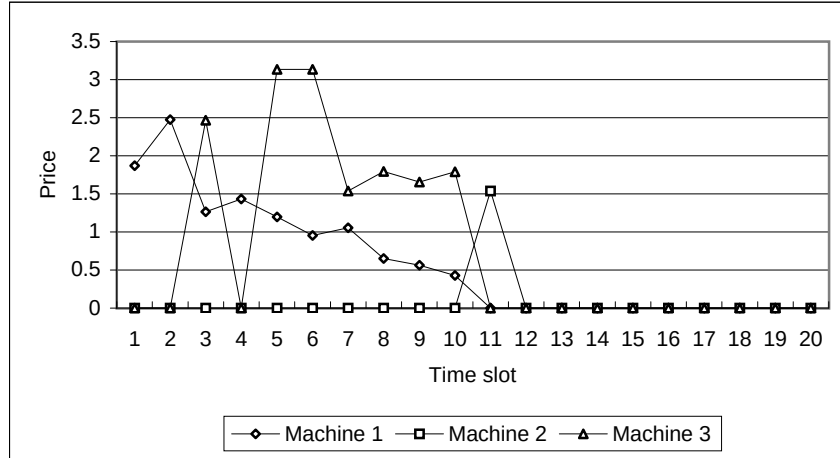


Figure 2.3: Price profiles at the end of *adaptive regular tâtonnement*

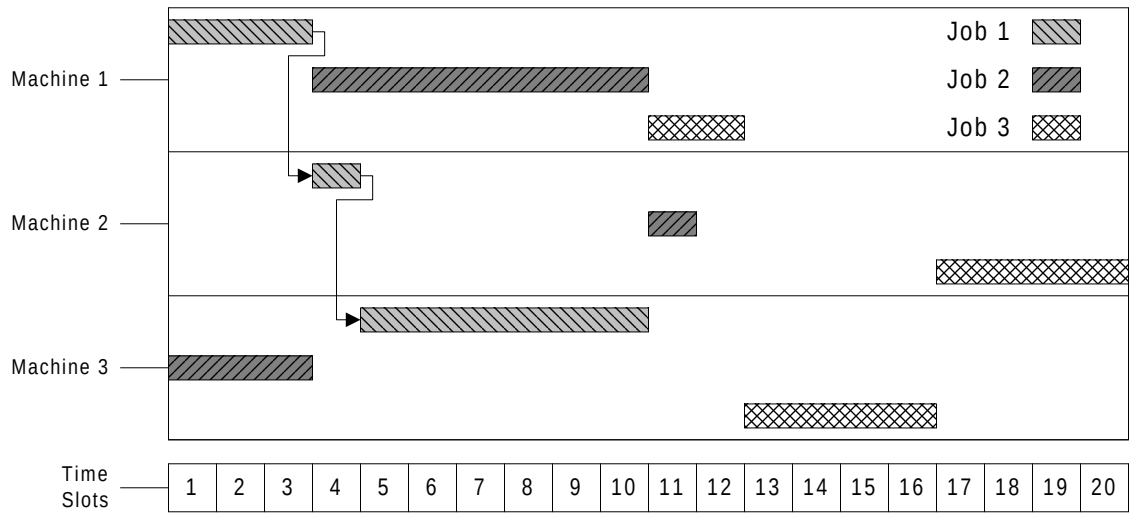


Figure 2.4: Feasibility-restored shop schedule after the fourth iteration (the best UB schedule which is also optimal).

2.4. EXAMPLES

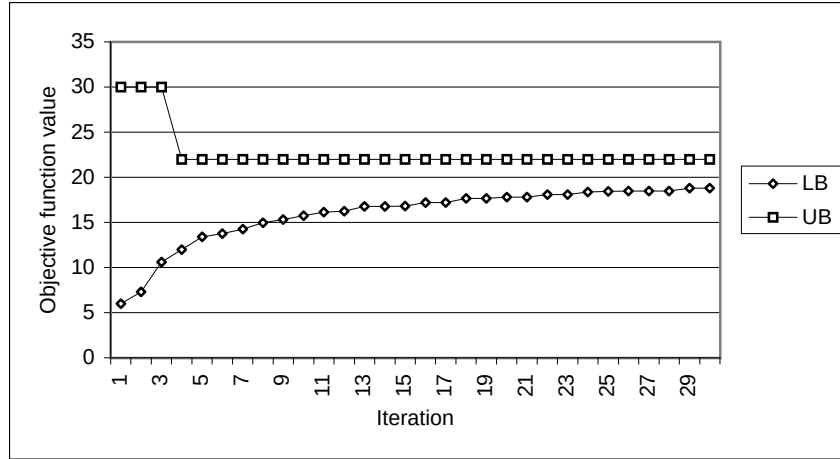


Figure 2.5: Lower bound (LB) and Upper bound (UB) during the *adaptive regular tâtonnement*

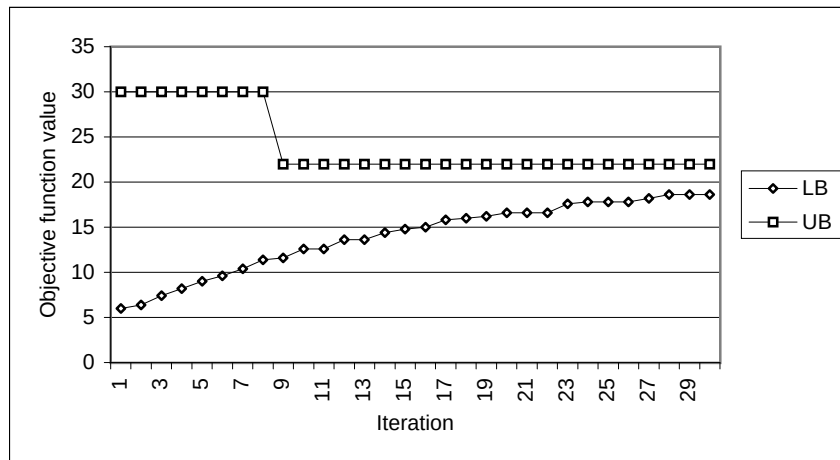


Figure 2.6: Lower bound (LB) and Upper bound (UB) values during the *non-adaptive regular tâtonnement*

2.4. EXAMPLES

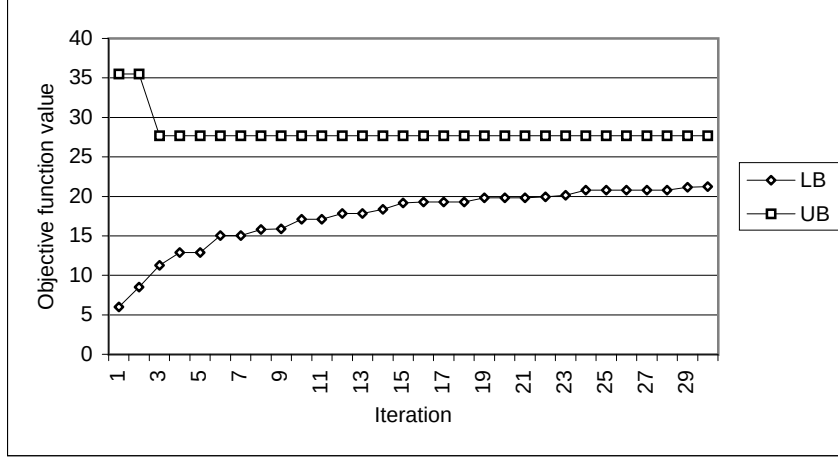


Figure 2.7: Lower bound (LB) and Upper bound (UB) values during the *adaptive augmented tâtonnement*

Table 2.3: 3x3 Bottleneck job shop example

Job	Weight	Due date	Machine (Processing time)
1	3	12	1 (7), 2 (3), 3 (2)
2	6	11	3 (3), 1 (6), 2 (2)
3	4	9	3 (1), 1 (5), 2 (3)

schedule shown in Figure 2.4) is found in iteration 3.

The implementation for *non-adaptive augmented tâtonnement* is straight-forward. Although there is no need to calculate lower and upper bound values during the tâtonnement, we keep track of the values to observe its convergence behavior and the quality of the final allocation (See Figure 2.8). The best feasible schedule is reached in iteration 7, and somewhat larger duality gap (32%) results at the end as compared to adaptive version.

2.4.2 Bottleneck Job Shop

This example is similar to the random job shop case except that there is a bottleneck machine (machine 1) where all jobs require a longer processing time (See Table 2.3).

2.4. EXAMPLES

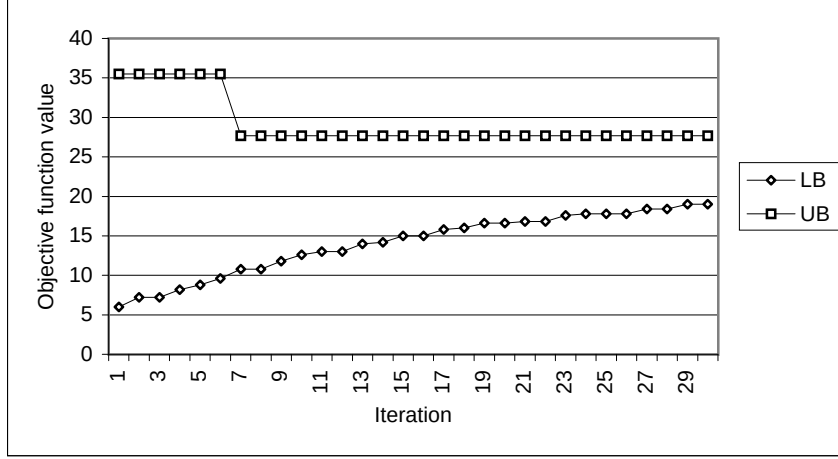


Figure 2.8: Lower bound (LB) and Upper bound (UB) during the *non-adaptive augmented tâtonnement*

Figure 2.9 shows the price profile at the end of *adaptive regular tâtonnement*. As the figure shows, almost all time slots, except the ones on the bottleneck machine, remain at their initial prices of zero. This should be of little surprise since the bottleneck machine is subject to the most competition among the job agents. Figure 2.10 shows that UB and LB improve significantly over the iterations reaching a final duality gap of only 2.6%. The UB schedule achieved in the 7th iteration is optimal with a weighted tardiness value of 54.

The implementation of *non-adaptive regular tâtonnement* yields similar results as summarized in Figure 2.11. The best feasible solution is found in iteration 10.

The *augmented tâtonnement* with the adaptive auction protocol yields results as shown in Figure 2.12. Using a perturbed payment function helps achieve the best upper bound in the second (2nd) iteration. The results for its non-adaptive counterpart are very similar to the adaptive version and are not presented.

2.4.3 Flow Shop

The flow shop data is shown in Table 2.4. Note that the jobs visit the machines in the same order but their processing times differ. The price profiles reached at the end of the *adaptive regular tâtonnement* are depicted in Figure 2.13. As one may expect, the

2.4. EXAMPLES

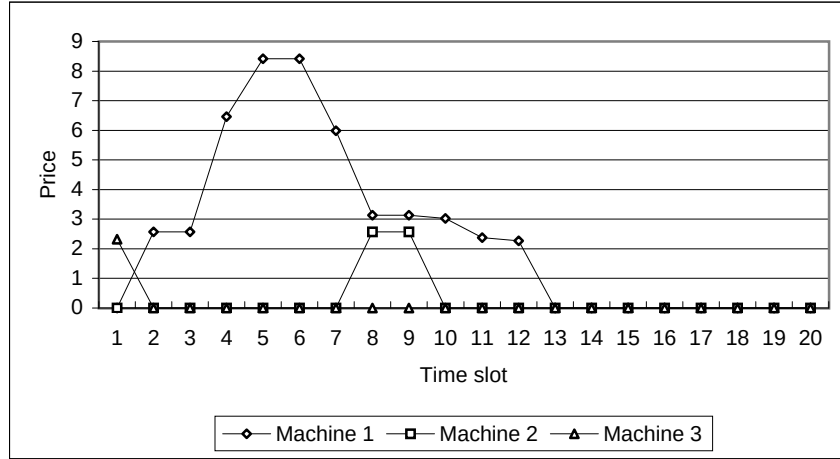


Figure 2.9: Price profiles for each machine at the end of *adaptive regular tâtonnement* (Bottleneck Job Shop)

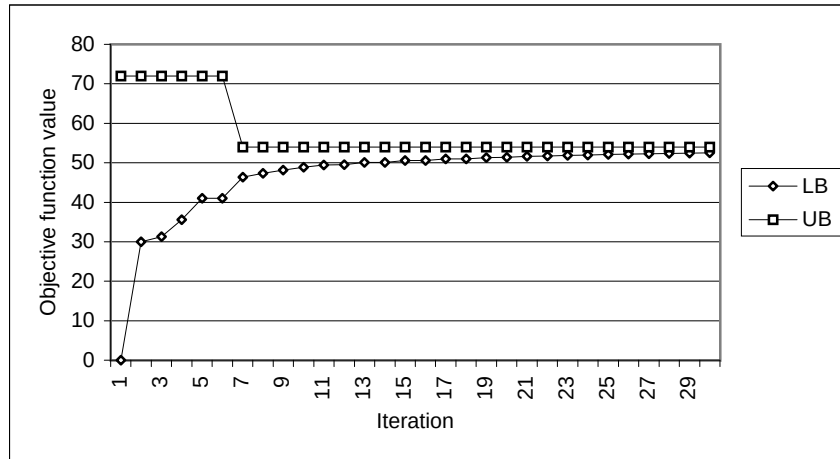


Figure 2.10: Lower bound (LB) and Upper bound (UB) values during the *adaptive regular tâtonnement* (Bottleneck Job Shop)

2.4. EXAMPLES

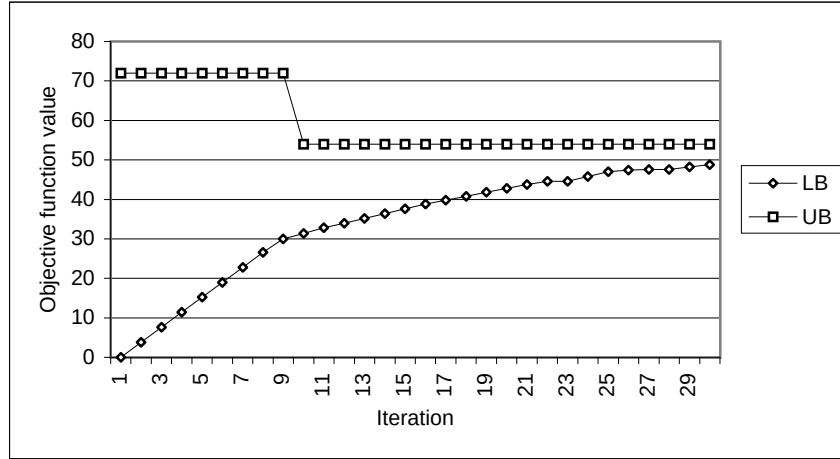


Figure 2.11: Lower bound (LB) and Upper bound (UB) values during the *non-adaptive regular tâtonnement* (Bottleneck Job Shop)

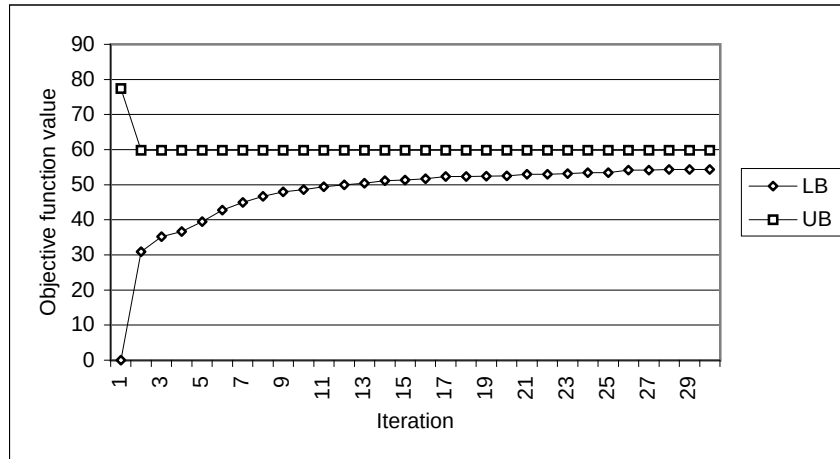


Figure 2.12: Lower bound (LB) and Upper bound (UB) values in *adaptive augmented tâtonnement* (Bottleneck Job Shop)

2.4. EXAMPLES

Table 2.4: 3x3 Flow shop example

Job	Weight	Due date	Machine (Processing time)
1	4	10	1 (3), 2 (1), 3 (6)
2	6	13	1 (3), 2 (7), 3 (1)
3	2	12	1 (2), 2 (4), 3 (4)

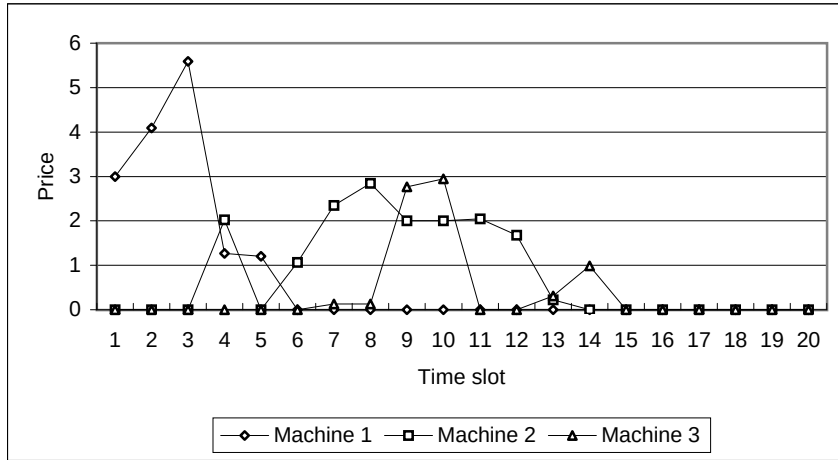


Figure 2.13: Price profiles at the end of *adaptive regular tâtonnement* (Flow Shop)

first machine has higher prices on the earlier time slots, while the second and third machines have positive prices in later time periods. As can be seen from Figure 2.14, the best upper bound (which is also optimal) is reached in third iteration, and the final duality gap is 17.2%. If we compare the duality gaps between different routing structures, we see that the bottleneck job shop has the smallest, the flow shop the next, and the job shop the largest.

Revised values of LB and UB for *augmented tâtonnement* with the adaptive auction protocol are depicted in Figure 2.15. It reaches the optimal solution in the second iteration. The results of the non-adaptive version of both the regular and the augmented tâtonnements are not presented for brevity, since they are consistent with the earlier observations (i.e. slower convergence, wider duality gaps).

2.4. EXAMPLES

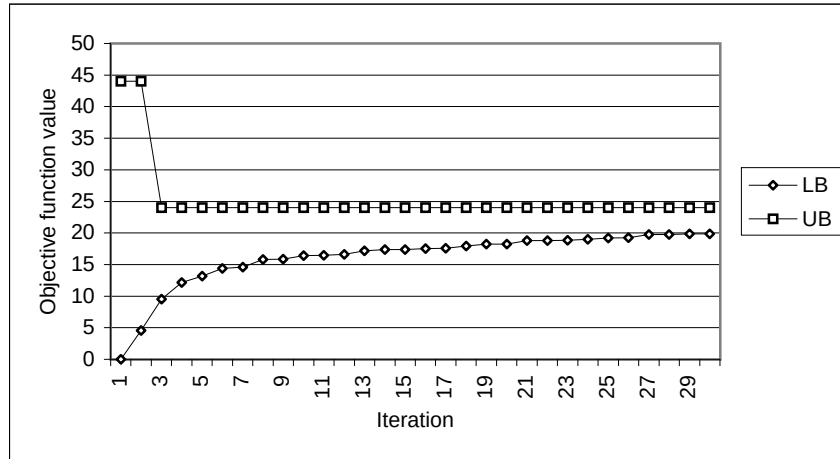


Figure 2.14: Lower bound (LB) and Upper bound (UB) values during the *adaptive regular tâtonnement* (Flow Shop)

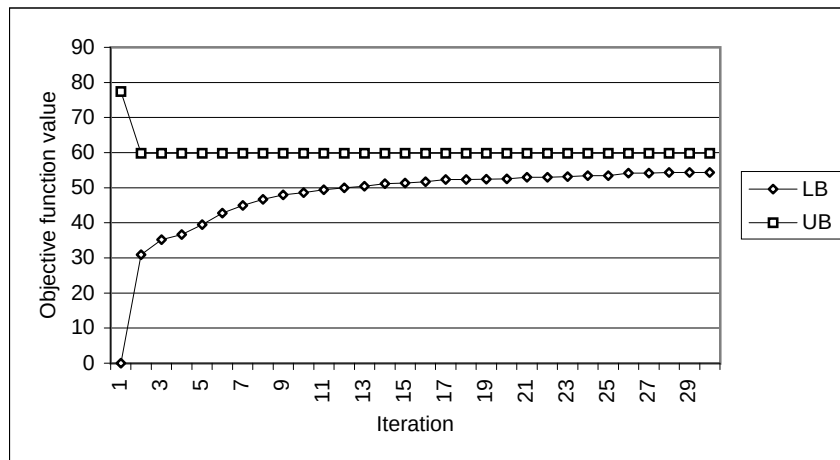


Figure 2.15: Lower bound (LB) and Upper bound (UB) during *adaptive augmented tâtonnement* (Flow shop)

2.5 Conclusions

In this chapter, we introduced a group of price-directed auction mechanisms for distributed scheduling. We investigated two *auction protocols* (non-adaptive Walrasian and adaptive tâtonnements) and two *payment functions* (regular and augmented tâtonnements). We showed that Lagrangean relaxation using subgradient search corresponds to an adaptive regular tâtonnement. We demonstrate the operations of these auction mechanisms using small examples representing three major routing structures (random job shop, bottleneck job shop, and flow shop). As demonstrated in the examples, the prices of time slots (objects) depend heavily on the demand patterns (i.e., routing structures and processing times) of job agents.

This chapter presents a step toward answering many important questions concerning distributed scheduling. By exploring the relationship among auction tâtonnements, discrete optimization algorithms, and distributed decision making, we are able to categorize different alternatives for implementation and their possible implications. The computational examples show that the adaptive price update with subgradient step might be superior to the non-adaptive auction protocol. Moreover, augmented tâtonnement using quadratic price discrimination might be effective in speeding up the convergence. Of course, a much more extensive computational study is necessary to generalize these results. Other generalization may include the cases where each agent responsible for several jobs. In this case, the agents' local problem will be more difficult to solve, but on the other hand, less resource conflicts may result.

In this study, we do not address the issue of incentive compatibility, i.e., how and why the job agents decide to participate in the auction and how to design incentives so that the job agents reveal their true preferences. We assume that all job agents will participate in the auction and will report their true optimal solutions. Earlier studies show that some price-directed mechanisms that guarantee the convergence to a globally optimal solution are incentive compatible in certain cases [19, 20]. There is no research to-date for the more complex discrete optimization problems. We believe that in order to achieve true distributed scheduling in realistic settings, the above issues and many other complicating factors need to be carefully addressed.

Chapter 3

Robustness in Distributed Scheduling

3.1 Introduction

We study scheduling robustness in a production environment where frequent changes in shop conditions are to be expected. These uncertain events complicate the role of production scheduling since two conflicting goals must be reconciled: first, the system must perform at a globally satisfactory level for efficient resource usages, and second, the system must allow sufficient local flexibility for changes and adjustments.

To achieve the first goal of high global performance a well-planned *a priori* schedule is required. In general, *static scheduling* techniques are used to optimize the operations schedule at the beginning of a planning horizon. This approach typically generates a detailed schedule assuming perfect information about the system's current and future states. However, unexpected disturbances make this pre-planned schedule very difficult, if not impossible, to follow. Moreover, overall system performance deteriorates when updates are made for the system changes.

To achieve the second goal of local flexibility, dynamic dispatching policies are typically used which make decisions as needed throughout the course of production. This *dynamic scheduling* approach typically uses myopic priority indices and greedy

3.1. INTRODUCTION

heuristics which may lead to poor global performance. However, its inherent flexibility and ability to utilize up-to-date local information makes *dynamic scheduling* a practical tool in uncertain environments.

In this study, we propose to address scheduling robustness by a method that reconciles the benefits of *static* and *dynamic* scheduling. We shift the focus of *a priori scheduling* to identifying decisions that are critical to global performance under the presence of random disturbances. These decisions are optimized using *a priori* stochastic information, which in turn preprocess, or partially solve the scheduling problem. The actual schedule unfolds over time using *dynamic scheduling*.

Treating scheduling problems in uncertain environments did not attract much attention in the literature until the early 80s. One approach is to reschedule all the jobs from scratch every time an uncertain event occurs [66, 99, 17, 3, 97, 98]. Researchers also proposed schedule repair methods that compute a temporary schedule after each machine breakdown with the attempt of bringing the system back to schedule in a finite amount of time. The “match-up scheduling” method proposed by Bean et al. [4] and Gallego [29, 30] belongs to this category. A similar schedule update heuristic that uses second moment information is proposed by Maddox and Birge [57].

Instead of repairing a pre-specified schedule, Mehta and Uzsoy [61] propose a method that insert idle times into the schedule in anticipation of schedule disruptions. Another approach is to split the scheduling activity into *planning* and *dispatching* stages. The planning stage computes the pricing or resource usage costs using global information, while the dispatching phase makes use of the resource pricing and up-to-date shop conditions and uncertainties. This approach is both general and effective as demonstrated by Roundy, et al. [77], and Morton et al. [63, 64]. A similar approach by Wu et al. [96] partitions the operations into an ordered sequence of subsets in the planning phase. This identifies and resolves the supposedly critical scheduling decisions through a graph-theoretic preprocessing. The second stage completes the schedule dynamically using a dispatching heuristic.

The approach described in this paper is in parallel to the idea of preprocessing scheduling problems using *a priori* information while dynamically generating the actual schedule.

3.2. PROBLEM STATEMENT AND ANALYSIS

Table 3.1: Notation

i	Job index, $i = 1, \dots, N$ where N is number of jobs
j	Operation index, $j = 1, \dots, n_i$ where n_i is number of operations of job i
k	Machine index, $k = 1, \dots, M$ where M is number of machines
t	Time index, $t = 1, \dots, T$ where T is number of time periods in planning horizon
W_i	Weight or tardiness penalty of job i
d_i	Due date of job i
C_i	Completion time of job i
T_i	Tardiness of job i , $\max\{0, C_i - d_i\}$
m_{ij}	Machine required for operation j of job i
p_{ij}	Processing time of operation j of job i
C_{ij}	Completion time of operation j of job i
S_{ij}	Start time of operation j of job i

3.2 Problem Statement and Analysis

3.2.1 The Mathematical Model

We consider the classical job-shop scheduling model where a set of jobs is to be completed on multiple machines. Each *job* consists of a series of *operations* that represent the production steps of the job. Due to processing requirements the operations of a job are subject to *precedence constraints*. Each production step (an operation) requires a specific *machine* and once the processing of an operation starts it can not be preempted by any other operations. The collection of operation *processing times* and their corresponding machine requirements represent the *job routing*. Each machine can only process one operation at a time. This requirement defines the *machine capacity constraints*. Each job has a *due-date* and a *weight*, and the objective is to find a schedule which minimizes the total weighted tardiness.

We first state an integer programming formulation of the job-shop scheduling problem due to Pritsker et al. [73]. The notation is summarized in Table 3.1. The

3.2. PROBLEM STATEMENT AND ANALYSIS

decision variable of this model (X_{ijt}) is defined as follows:

$$X_{ijt} = \begin{cases} 1 & \text{if operation } j \text{ of job } i \text{ has started by time } t, \\ 0 & \text{otherwise.} \end{cases} \quad (3.1)$$

The JSP then can be formulated as follows (we set the variables with out-of-range indices to zero):

(JSP)

$$\min \sum_i W_i \left[\sum_{t > d_i - p_{i,n_i}} (1 - X_{i,n_i,t}) \right] \quad (3.2)$$

$$\text{s.t. } X_{i,j,t+1} \geq X_{i,j,t}, \quad \forall i, j, t < T \quad (3.3)$$

$$X_{i,j,t} \leq X_{i,j-1,t-p_{i,j-1}}, \quad \forall i, j > 1, t \quad (3.4)$$

$$\sum_{i,j:m_{ij}=k} (X_{i,j,t} - X_{i,j,t-p_{ij}}) \leq 1, \quad \forall k, t \quad (3.5)$$

$$X_{i,j,t} \in \{0, 1\}, \quad \forall i, j, t \quad (3.6)$$

The first set of constraints (3.3) makes sure that once an operation is started, it remains so in all subsequent time periods. This is due to the definition of X_{ijt} and the non-preemption requirement. Constraints (3.4) state that an operation cannot start until all its predecessors are completed. Finally, the machine capacity constraints (3.5) state that at most one job can be processed on a particular machine in a given time period. The objective function (3.2) is the total weighted tardiness derived from the fact that a cost of W_i is incurred for each time period after d_i during which job i has not been completed.

The objective function can be rewritten as follows:

$$\sum_i W_i \left[\sum_{t > d_i - p_{i,n_i}} (1 - X_{i,n_i,t}) \right] = \sum_i W_i (T - d_i + p_{i,n_i}) + \sum_i \sum_j \sum_t -w_{ijt} X_{ijt} \quad (3.7)$$

where

$$w_{ijt} = \begin{cases} W_i & \text{if } j = n_i \text{ and } t > d_i, \\ 0 & \text{otherwise.} \end{cases} \quad (3.8)$$

3.2. PROBLEM STATEMENT AND ANALYSIS

Since the first term in the objective function (3.7) is a constant we represent it as $A = \sum_i A_i$ where $A_i = W_i(T - d_i + p_{i,n_i})$.

In solving scheduling problems we know that the processing time information is among the most critical input. This is particularly true when a *regular* performance measure such as (3.2) is used. Almost all successful dispatching heuristics rely heavily on processing time information. In our IP formulation, both objective function and constraints are defined based on processing times. Any change in processing times will affect the solution and its corresponding weighted tardiness value. Since processing times are typically uncertain in practice, best estimates or expected values are used in deterministic scheduling models. This common treatment of processing time information ignores the effects of processing time variations, which could clearly lead to poor quality schedules in the face of uncertainty. We will address these important issues in the following sections.

3.2.2 The Proposed Scheduling Scheme and Broader Managerial Insights

Before stating the scheduling scheme, we first make two observations from the industry scheduling environments: (1) at the time of planning a schedule there is typically some information available concerning future state of the shop, and (2) there is no need for a completely specified schedule for all jobs over the entire planning horizon. Using the available information, a seasoned scheduler can focus his/her attention on critical decisions and typically pinpoint a few of those to achieve overall schedule efficiency. Detailed scheduling decisions are often delayed until actual production when up-to-date information is available with much higher certainty.

Motivated by this practical paradigm of decision making we propose a few adaptations to the IP model that changes the role it plays in the scheduling process. Instead of using the (JSP) model to generate detailed static schedules, we use it to analyze the criticality of scheduling decisions based on *a priori* information. Specifically, we modify (JSP) to include *chance constraints*, which captures the effects of processing time variations. After a Lagrangean relaxation of the machine capacity constraints,

3.2. PROBLEM STATEMENT AND ANALYSIS

the chance-constrained program has a special network structure in its subproblems. The solution to this relaxed problem is a partially solved scheduling problem (a lower bound) since many resource conflicts are left unresolved. To improve the lower bound we start iterating on the relaxed problem using a subgradient search algorithm. The algorithm stops when the partial schedule contains sufficient details to be released to the shop for planning purposes (or when the computing budget runs out). We leave the remainder of the scheduling decisions to system dynamics, i.e., we resolve the remaining resource conflicts as needed during the actual implementation of the schedule.

The proposed scheduling scheme can be considered a version of the preprocess-first-schedule-later (PFSL) scheme proposed by Wu et al. [96]. The preprocessing is performed by the Lagrangean-relaxed, chance-constrained mathematical program, while the detailed scheduling is carried out by dynamic dispatching. The PFSL scheme is based on the following conjecture:

Conjecture 1: The robustness of *static scheduling* model such as (JSP) can be significantly improved by preprocessing the problem instance using *a priori* analysis of its structure, while making detailed scheduling decisions dynamically. The extent of the improvement depends on the underlying statistical distribution and the level of uncertainty.

This conjecture has been strongly supported by the experiments in [96] where they use a branch and bound algorithm for the *a priori* analysis and a dynamic dispatching rule for detailed scheduling. After testing under uniform and exponentially distributed variations, they show that the marginal value of *a priori* analysis deteriorates as the level of uncertainty increases. Not surprisingly, their preprocessing scheme uses only the structure of the deterministic model, i.e. no stochastic information is used.

Knowing that *a priori* analysis of problem structure may improve scheduling robustness, the goal of the current study is to improve scheduling robustness using *a priori* stochastic information. A question of general managerial interest is “what are the insights one could extract from problem structure in the context of *a priori*

3.3. SOLUTION METHODOLOGY

stochastic information, and how do they improve scheduling robustness under uncertainty.” To address this broader research question, we state three main conjectures which our proposed scheme builds on. These conjectures will be thoroughly tested in the computational experiments.

Conjecture 2: Scheduling robustness can be significantly improved by incorporating not only the mean but also the second moment of the stochastic information. Moreover, the value of this information increases as the level of uncertainty increases.

Conjecture 3: The robustness of *dynamic dispatching* can be significantly improved by *a priori* static analysis using stochastic information. The extent of the improvement varies by problem instances, the type of underlying distributions, and the level of uncertainty.

Conjecture 4: The marginal benefit of *a priori* stochastic analysis (to the PFSL scheme) increases as the level of uncertainty increases, and its effect varies by the type of underlying distribution.

Testing these conjectures provides support (and counter cases) for the belief that one could reconcile the benefits of static analysis and dynamic adaptation, while making effective use of *a priori* stochastic information.

3.3 Solution Methodology

We will start constructing our solution methodology applying Lagrangean relaxation to the JSP formulation as suggested by [77]. We first relax machine capacity constraints (3.5), and decompose the problem into job-level subproblems with integrality property. We then employ chance-constrained programming at the subproblem level which incorporates *a priori* stochastic information.

3.3. SOLUTION METHODOLOGY

3.3.1 Lagrangean Relaxation

When we dualize the machine capacity constraints (3.5) of JSP with nonnegative Lagrangean multipliers $\lambda_{kt}, \forall k, t$ we obtain the following Lagrangean Relaxation problem (LR):

(LR_λ)

$$\min \sum_i A_i + \sum_i \sum_j \sum_t (\lambda_{m_{ij},t} - \lambda_{m_{ij},t+p_{ij}} - w_{ijt}) X_{ijt} - \sum_k \sum_t \lambda_{kt} \quad (3.9)$$

$$\text{s.t. } X_{i,j,t+1} \geq X_{i,j,t} \quad \forall i, j, t < T \quad (3.10)$$

$$X_{i,j,t} \leq X_{i,j-1,t-p_{i,j-1}} \quad \forall i, j > 1, t \quad (3.11)$$

$$X_{i,j,t} \in \{0, 1\} \quad \forall i, j, t \quad (3.12)$$

We decompose LR_λ into independent job-level subproblems as follows:

$$v(LR_\lambda) = \sum_i v(LR_{\lambda,i}) - \sum_k \sum_t \lambda_{kt} \quad (3.13)$$

where $v(P)$ denotes the value of optimal solution of problem P , and

$$v(LR_{\lambda,i}) = \min A_i + \sum_j \sum_t (\lambda_{m_{ij},t} - \lambda_{m_{ij},t+p_{ij}} - w_{ijt}) X_{ijt} \quad (3.14)$$

$$\text{s.t. } X_{i,j,t+1} \geq X_{i,j,t} \quad \forall j, t < T \quad (3.15)$$

$$X_{i,j,t} \leq X_{i,j-1,t-p_{i,j-1}} \quad \forall j > 1, t \quad (3.16)$$

$$X_{i,j,t} \in \{0, 1\} \quad \forall j, t \quad (3.17)$$

The subproblems satisfy integrality property, in fact they are dual of a specially structured maximum flow network problem [77]. We can therefore relax the binary constraints (3.17) as below:

$$X_{ijt} \leq 1, \quad \forall j, t \quad (3.18)$$

3.3. SOLUTION METHODOLOGY

$$X_{ijt} \geq 0, \forall j, t \quad (3.19)$$

Repeated application of first set of constraints (3.15) leads us to the following constraints instead of relaxed constraints (3.18), (3.19):

$$X_{ijT} \leq 1, \forall j \quad (3.20)$$

$$X_{ijt} \geq 0, \forall j, t \quad (3.21)$$

Since the dual to $(LR_{\lambda,i})$ is a max-flow problem on a network, we may use special dual network flow algorithms to solve each job-level subproblem.

3.3.2 Chance-Constrained Programming

We will now construct a chance-constrained model for each job-level subproblem by first accepting the fact that the processing times are uncertain. We assume that we have information about the type of processing time distribution, and its mean and variance. This information is typically easy to estimate in a real system using historic data.

Chance-constrained programming was first proposed by Charnes and Cooper [14]. In this approach, we use chance-constraints to represent processing time variation instead of using deterministic constraints that assume expected processing times. For example, if we want to represent a constraint, say $x < b$ (b is not perfectly known), that holds with a probability of at least β , the corresponding chance-constraint is $P[x < b] \geq \beta$ where $P[\cdot]$ is probability function for b . In job-level subproblems, precedence constraints are the only constraints that are affected by processing time uncertainty. For a specific job i , the precedence constraints are as follows:

$$X_{ijt} \leq X_{i,j-1,t-p_{i,j-1}}, \forall j > 1, t \quad (3.22)$$

Since processing times p_{ij} occur in indices in (3.22), it is difficult to convert these constraints as they are. However, we can obtain start and completion times (S_{ij} and

3.3. SOLUTION METHODOLOGY

C_{ij}) of each operation as follows:

$$S_{ij} = T - \sum_t X_{ijt} + 1 \text{ and } C_{ij} = T - \sum_t X_{ijt} + p_{ij} \quad (3.23)$$

Hence we can rewrite the precedence constraints (3.22) as follows:

$$C_{ij} \leq S_{i,j+1} - 1, \forall j < n_i, \quad (3.24)$$

or equivalently

$$\sum_t X_{ijt} - \sum_t X_{i,j+1,t} \geq p_{ij}, \forall j < n_i \quad (3.25)$$

We can now incorporate probabilistic processing times as follows: Suppose we want a *confidence level* of β_{ij} for each operation j of job i to satisfy its stated precedence constraint (3.25), then we have the following chance constraints:

$$P \left[\sum_t X_{ijt} - \sum_t X_{i,j+1,t} \geq p_{ij} \right] \geq \beta_{ij}, \forall j < n_i \quad (3.26)$$

Further, instead of chance constraints (3.26), we substitute the following deterministic and equivalent constraints:

$$\sum_t X_{ijt} - \sum_t X_{i,j+1,t} \geq P_{ij}, \forall j < n_i \quad (3.27)$$

where P_{ij} is the smallest number satisfying

$$P[p_{ij} \leq P_{ij}] \geq \beta_{ij} \text{ or } F[P_{ij}] \geq \beta_{ij} \quad (3.28)$$

where $F[\cdot]$ is cumulative distribution function. Specifically, if the processing time p_{ij} is known to be uniformly distributed between a_{ij} and b_{ij} , then the corresponding *adjusted processing time* $P_{ij} = a_{ij} + \beta_{ij}(b_{ij} - a_{ij})$. If p_{ij} is predicted to be normally distributed with $N(\bar{p}_{ij}, \sigma_{ij}^2)$, then $P_{ij} = \bar{p}_{ij} + K_{\beta_{ij}} \sigma_{ij}$ where $K_{\beta_{ij}}$ is standard normal value corresponding to β_{ij} .

We can now convert the chance-constraint equivalents into their original form

3.3. SOLUTION METHODOLOGY

with the *adjusted processing times* included in indices. This results in a job-level subproblem (referred as $LRC_{\lambda,i}$ for LR with *chance-constraints*) that minimizes the subproblem objective defined in (3.14) subject to chance-constrained precedence, non-preemption constraints and bound constraints for the decision variables. The job level subproblem with chance constraints for job i can be stated as follows:

$$v(LRC_{\lambda,i}) = \min A_i + \sum_j \sum_t (\lambda_{m_{ij},t} - \lambda_{m_{ij},t+P_{ij}} - w_{ijt}) X_{ijt} \quad (3.29)$$

$$\text{s.t. } X_{i,j,t} \leq X_{i,j-1,t-P_{i,j-1}} \quad \forall j > 1, t \quad (3.30)$$

and constraints (3.15), (3.20), (3.21).

The revised Lagrangean objective is:

$$v(LRC_{\lambda}) = \sum_i v(LRC_{\lambda,i}) - \sum_k \sum_t \lambda_{kt} \quad (3.31)$$

We can still solve each job-level subproblem using a dual network algorithm because the structure of the current model is the same as the deterministic version. We use a *subgradient search algorithm* for the Lagrangean relaxation problem with chance constraints. Our subgradient optimization starts with an initial value for the multipliers $\lambda^0 = 0$. Then the algorithm generates a sequence $\{\lambda^r\}$ over the iterations r by the rule

$$\lambda_{kt}^{r+1} = \max\{0, \lambda_{kt}^r + s_r g_{kt}(X^r)\} \quad (3.32)$$

where s_r is an appropriately selected step size and $g_{kt}(X^r)$ is the subgradient of the capacity constraint of machine k for time period t defined by the optimal solution X^r of LRC_{λ^r} :

$$g_{kt}(X^r) = \sum_i \sum_{j:m_{ij}=k} (X_{i,j,t}^r - X_{i,j,t-P_{ij}}^r) - 1 \quad (3.33)$$

Step size s_r is calculated as follows:

$$s_r = \alpha_r \left(\frac{UB - v(LRC_{\lambda^r})}{\sum_k \sum_t g_{kt}^2(X^r)} \right) \quad (3.34)$$

where α_r is a scalar satisfying $0 < \alpha_r \leq 2$ and UB is a target upper bound value

3.3. SOLUTION METHODOLOGY

for *JSP* which can be updated over the iterations. Details of the subgradient search algorithm is given in Section 3.3.4.

3.3.3 Scaling, Lagrangean Ranking and Schedule Preprocessing

Before going into further details of the subgradient search algorithm, note that in the JSP formulation, the number of discrete time periods T has a significant effect on the computational tractability of the model. We scaled the processing times by a *scaling factor* which provides an upper bound to the original model at a reduced computational cost. In addition to reducing computer time, the scaling factor results in several interesting additional benefits. First, since the JSP model is used to preprocess (rather than solve) the scheduling instance at hand, we are more interested in analyzing the aggregate structure of the problem rather than the minute-to-minute details. The *scaling factor* allows us to adjust the “resolution” of the analysis. Second, since we consider uncertain processing times, the stochastic information about processing times could be used to determine a proper scaling factor. Third, for regular performance measures (such as weighted tardiness) a unique schedule can be determined by the operation ranking in each machine. For example, one does not need to know the actual processing times to implement an SPT dispatching rule. Knowing the relative processing time ranking would suffice. Thus, in theory, one could maximize the scale of the processing time so long as the operational ranking is still distinguishable. In general, the *scaling factor* provides a parametric trade-off between model resolution and computational efficiency. Throughout our implementation, we use scaled, adjusted processing times, P_{ij}^s for the model.

Another important aspect of our proposed scheme is that we use Lagrangean Relaxation and subgradient search to find a partially established ranking among the operations. Specifically, in each subgradient iteration, we combine all solutions from job-level subproblems to get a capacity-infeasible shop schedule. Based on this *Lagrangean Schedule*, we define *Lagrangean Ranking* as follows:

3.3. SOLUTION METHODOLOGY

Definition: (*Lagrangean Ranking*) Consider three specific conditions between operation pairs (i, j) and (i', j') in the *Lagrangean Schedule* as follows:

- (1) (i, j) and (i', j') require the same machine, i.e., $m_{ij} = m_{i'j'}$,
- (2) there is no resource conflict between (i, j) and (i', j') in the *Lagrangean Schedule*, i.e., $(X_{i,j,t} - X_{i,j,t-P_{ij}}) + (X_{i',j',t} - X_{i',j',t-P_{i'j'}}) \leq 1$, and
- (3) operation (i, j) starts earlier than (i', j') in the *Lagrangean Schedule*, i.e., $S_{ij} \leq S_{i'j'}$

If an operation pair (i, j) and (i', j') satisfies (1)-(3) then we say that it is *fully ranked*, and operation (i, j) has a higher priority than (i', j') . For operation pairs which satisfy conditions (1) and (3) but not (2), we say that they are *weakly ranked*. All the other operation pairs are *unranked*.

It is clear from the definition that only operation pairs with no resource conflict in the *Lagrangean Schedule* are *fully ranked*. Using the *Lagrangean Ranking* we can now define a preprocessed scheduling instance by imposing a precedence constraint (3.4) between each *fully ranked* pair of operations but not the *weakly ranked* pairs. Thus, the *weakly ranked* pairs of operations represent scheduling decisions which remain to be made. Note that under dynamic dispatching, the Lagrangean ranking only takes effect when the ranked pair (i, j) and (i', j') are both in the candidate set. When they do not appear in the candidate set at the same time, the Lagrangean ranking is relaxed. This is necessary in order to maintain the semi-active scheduling property [65]. Suppose we apply an active or non-delay schedule generation routine on this preprocessed scheduling instance using any priority dispatching rule, a feasible upper bound (*UB*) solution for JSP can be generated. In this context the priority specified by the *weak ranking* could be used for job dispatching. Alternatively, one could use a dynamic dispatching rule such as the Apparent Tardiness Cost (ATC) rule [90] which makes use of dynamic shop information over time.

The procedure described above can be used to generate upper bounds during each Lagrangean iteration. More importantly, we will use the procedure as a means to implement the Preprocess-First-Schedule-Later scheme described earlier. This PFSL scheme is outlined as follows:

3.3. SOLUTION METHODOLOGY

1. At the beginning of the planning period, use *a priori* scheduling data and stochastic information to formulate a chance-constrained, Lagrangean-relaxed scheduling problem (as described in Sections 3.3.1 and 3.3.2).
2. Solve the stated scheduling problem using a subgradient search algorithm (Section 3.3.4). In each iteration, compute LB using the *Lagrangean Schedule*, compute UB using the associated *Lagrangean Ranking*. At the end of the subgradient search, the *Lagrangean Ranking* associated with the best UB is returned for preprocessing.
3. Preprocess the scheduling instance, i.e., adding precedence constraints for each *fully ranked* operation pairs.
4. Release the preprocessed partial schedule to the shop. Generate the detailed schedule dynamically using priority dispatching.

3.3.4 The Subgradient Search Algorithm

The subgradient search algorithm is summarized in Figure 3.1. When we implement the algorithm, we generate the initial upper bound (Step 3) using the Apparent Tardiness Cost (ATC) rule [90]. ATC is shown to be an effective heuristic for the weighted tardiness objective [44, 45]. We implemented a *non-delay* and an *active* version of the ATC heuristic. The Lagrange multipliers are initialized at zero. In computing the upper bound at each subgradient iteration (Step 5.2) we implemented a non-delay and an active version using priority dispatching defined by (1) *weak ranking*, or (2) the ATC rule. Results of all four combinations are reported in the computational testing.

Another important implementation choice is the use of scaled or unscaled processing time for upper bound calculation. Clearly the upper bound UB_r used for step size calculation in Steps 5.2-5.5 must be computed from scaled processing times since it should be consistent with the lower bound values. However, in Steps 5.3 and 6 when we try to determine the “best” upper bound, we could use an upper bound generated with the unscaled processing times. This is because the unscaled

3.3. SOLUTION METHODOLOGY

- Step 1. Read input data: number of jobs N , number of machines M , job weights W_i , due dates d_i , mean (unscaled) processing times p_{ij} , machine requirements m_{ij} . Read distribution information (a_{ij} and b_{ij} in the Uniform case, σ_{ij}^2 in the Normal case).
- Step 2. Calculate adjusted processing times P_{ij} for constraints (3.30) according to the distribution type and confidence level β . Scale the adjusted processing times to obtain scaled processing times P_{ij}^s . Scale the due dates accordingly.
- Step 3. Upper bound initialization: generate an active or a non-delay schedule using a dispatching rule and scaled processing times. Use the upper bound schedule to set a reasonable planning horizon T .
- Step 4. Initialize the Lagrange multipliers λ_{kt} , the iteration counter r , and the scalar parameter α_r .
- Step 5. Main Step: Perform the subgradient search for K iterations.
 - Step 5.1 Compute the Lagrangean lower bound (LB_r): solve the job-level subproblems (3.29), (3.30), (3.15), (3.20), (3.21) using scaled processing times P_{ij}^s . This results in a capacity-infeasible schedule with objective value $v(LRC_{\lambda^r})$ which provides a lower bound for iteration r .
 - Step 5.2 Compute the Lagrangean upper bound (UB_r): Determine the *Lagrangean Ranking* from the lower bound schedule. Preprocess the scheduling instance using the *fully ranked* operation pairs, i.e., add a precedence constraint for each *fully ranked* pairs. Use a non-delay or an active schedule generation routine and a specified priority dispatching rule, generate a feasible schedule from the preprocessed scheduling instance. This provides an upper bound for iteration r .
 - Step 5.3 If the upper bound value UB_r is lower than the best UB so far, then update the upper bound, $UB \leftarrow UB_r$. Record the current *Lagrangean Ranking*.
 - Step 5.4 Update scalar α_r if necessary.
 - Step 5.5 Compute the subgradient $g_{kt}(X^r)$, step size s_r and update the Lagrangean multipliers λ_{kt}^{r+1} based on (3.33), (3.34), and (3.32), respectively.
- Step 6. Report the *Lagrangean Ranking* corresponding to the best UB over the K iterations (or alternatively, report the *Lagrangean Ranking* corresponding to the best lower bound *Lagrangean Schedule*). This is the *Lagrangean Ranking* to be released for schedule preprocessing.

Figure 3.1: The Subgradient Search Algorithm

3.4. COMPUTATIONAL STUDY

processing times may provide a more accurate assessment of the ultimate scheduling performance given a particular *Lagrangian Ranking*. Both implementations are reported in the computational study.

3.4 Computational Study

3.4.1 Experimental Design

We tested the proposed solution methodology in a Monte-Carlo environment simulating shop disturbances. The base test problems are generated from standard makespan job shop scheduling problems as described in [12]. We use 5 10x10 (10 jobs, 10 machines), 3 20x15, and 3 30x10 problems for testing. We coded each job-level scaled subproblem with chance-constraints in AMPL [27] and solved using dual network simplex algorithm in CPLEX [68]. A Pentium Pro 200 MHz personal computer with 64MB of RAM is used for all testing. We set initial parameter values as follows: $\lambda_{kt}^0 = 0 \forall k, t$, and $\alpha_0 = 2$ for the subgradient optimization procedure. We halve the value of α if there is no improvement in the Lagrangian lower bound for 3 iterations and set the total number of iterations K at 100 for 10x10 problems, and 200 for the others. For scaling, we divided adjusted processing times by a fixed value and rounded them to nearest integer values to obtain *scaled processing times* between 1 and 10. We scaled the due dates similarly. We used a common confidence value β for all chance-constraints. We tested two confidence levels as $\beta = 50\%$, and $\beta = 80\%$. Note that 50% confidence corresponds to using only mean (original) processing times whereas 80% confidence uses the variance information.

To simulate the stochastic environment, we generate actual processing times p'_{ij} using two distribution types: (1) *Uniform*: $U[a_{ij}, b_{ij}]$ where $p_{ij} = \frac{a_{ij} + b_{ij}}{2}$, and (2) *Normal*: $N[p_{ij}, \sigma_{ij}^2]$. Note that we assume the processing times p_{ij} specified in the scheduling data is the mean of the underlying processing time distribution. We tested three levels of processing time variation (V): 10%, 30% and 50%. In the case of Uniform distribution, the range parameters are calculated as $a_{ij} = p_{ij} - V * \bar{p}$ and $b_{ij} = p_{ij} + V * \bar{p}$ where $\bar{p}$ is the overall mean of the processing times. Thus, actual

3.4. COMPUTATIONAL STUDY

Table 3.2: Experimental factors and their levels in the simulation experiments (Each factor combination is run 10 replications).

Factor	Levels	No of Levels
Problem Size	10x10, 20x15, 30x10	3
Confidence Level	(50%), (80%)	2
Distribution Type	Uniform, Normal	2
Uncertainty Level	Low (10%), Medium (30%), High (50%)	3

processing times p'_{ij} are generated using a discrete uniform distribution $U[a_{ij}, b_{ij}]$. For the Normal distribution case, we equalize the variance with the Uniform case as follows, i.e. we set

$$\sigma_{ij}^2 = \frac{(b_{ij} - a_{ij})^2}{12}$$

where b_{ij} and a_{ij} are from the Uniform case with corresponding variation level V . Note that since a_{ij} and b_{ij} are calculated using single overall mean, the Normal variance values are equal for all the operations, i.e. $\sigma_{ij}^2 = \sigma^2 = V^2 \bar{p}^2 / 3$. We truncated negative numbers and 0 generated in the implementation of these formulas to 1. For a summary of experimental factors, see Table 3.2.

We test the performance of the proposed scheduling scheme under stochastic disturbances using the following experimental steps:

Step 1. Given a scheduling problem and the distribution information about the processing times, use the proposed PFSL scheme to find a preprocessed partial schedule using the *Lagrangian Ranking*.

Step 2. Complete the remainder of scheduling decisions over time using a specified dispatching method (i.e., Weak Ranking or ATC-ranking). Simulate this process using a Monte Carlo experiment where the operation processing times vary according to the specified distribution. At each dispatching point, the dispatching method uses actual processing time that are known up to point while using the original (mean) processing time for future operations. Report the total weighted tardiness at the end of the simulation.

3.4. COMPUTATIONAL STUDY

Table 3.3: Experimental results with ATC dispatching (Each cell shows the average total weighted tardiness over 50 instances: 5 10x10 problems each with 10 replications)

	Mean-Value Information		Perfect Information	
	Active	Nondelay	Active	Nondelay
Variation Level	Variation Type: Uniform Distribution			
10%	8941.90	6417.16	8508.92	6497.04
30%	8847.56	7277.32	9058.54	7223.88
50%	9417.68	8052.56	10452.96	7969.96
Variation Level	Variation Type: Normal Distribution			
10%	8905.22	6470.26	8865.3	6595.34
30%	8543.24	6951.28	8663.44	6767.08
50%	8373.46	7239.32	9454.66	6960.24

3.4.2 The Dynamic Dispatching Benchmark

To establish a benchmark for scheduling robustness we use the ATC dynamic dispatching heuristic. As demonstrated in previous research [96, 12], dynamic dispatching heuristics such as ATC serve as an excellent benchmark since they provide consistent performance under a wide variety of shop conditions. To provide a broader base for the benchmark we implemented four different versions of the ATC heuristic using *active* and *non-delay* schedule generation routines [72] assuming *perfect* and *mean-value* information on the processing times. In the *perfect information* case, we first generate the actual (after perturbation) processing times p'_{ij} and make this information available for each ATC priority calculation. In the *mean-value* case the ATC heuristic uses the mean processing time provided with the scheduling problem for future jobs and actual processing times for jobs that are already dispatched. In both cases, the heuristic is evaluated using Monte-Carlo simulation with Uniform and Normal processing-time perturbation described above. In Table 3.3 we first summarize the results of the four different ATC implementations. As shown in the table, ATC appears to be quite robust under different processing time variations. The heuristic yields better performance in the normal variation case than in the uniform case at the same variation level. While non-delay ATC schedules significantly outperform their

3.4. COMPUTATIONAL STUDY

Table 3.4: Experimental results of the PFSL methodology (Each cell shows the average total weighted tardiness over 50 instances: 5 10x10 problems each with 10 replications).

<i>UB Calc.</i>	Unscaled Processing times				Scaled Processing Times			
<i>Dispatching</i>	Weak Ranking		ATC-Ranking		Weak Ranking		ATC-Ranking	
<i>Sch. Routine</i>	Active	Nondel.	Active	Nondel.	Active	Nondel.	Active	Nondel.
β, V	<i>Variation Type: Uniform Distribution</i>							
80%, 10%	4566	6327	4980	6015	4973	6186	5323	5722
80%, 30%	5165	7170	5911	6917	5413	7037	6080	7191
80%, 50%	6627	8163	6959	8561	7714	8585	8011	8286
50%, 10%	4489	6181	5525	6839	5079	7500	5519	6956
50%, 30%	5879	7694	6964	7650	5853	7908	6481	7772
50%, 50%	8042	9094	9020	9019	7995	9347	8165	9060
β, V	<i>Variation Type: Normal Distribution</i>							
80%, 10%	4318	6557	5380	6110	4660	6839	5940	5869
80%, 30%	5431	6945	6121	6720	5942	6861	5829.60	6587.56
80%, 50%	6013	7721	6737	7887	6590	8080	6283	7258
50%, 10%	4630	6198	5299	6358	4969	6969	5552	7004
50%, 30%	5516	6865	6551	6738	6058	7517	6412	7129
50%, 50%	6792	7759	7913	7237	7448	8154	7412	7828

active counterparts, the difference between the overall performance of perfect and that of mean-value information is insignificant (sometimes, perfect information leads to poorer performance). This non-intuitive result can be explained by the fact that the ATC index makes look ahead approximation at each dispatching point using aggregated future information on each job. As a result, using mean or actual processing times may not yield significant performance difference. Based on this result, we will compare our approach with non-delay ATC procedure using mean-value processing times.

3.4.3 Testing Implementation Choices

To test the proposed PFSL scheme using Lagrangean preprocessing we first compare the effects of three different implementation choices using 10x10 test problems:

- *Upper Bound Calculation*: use unscaled vs. scaled processing time for upper bound selection and schedule evaluation (see Section 3.3.4),

3.4. COMPUTATIONAL STUDY

- *Dynamic Dispatching*: After the Lagrangean preprocessing, complete the schedule dynamically using *weak-ranking* vs. ATC-ranking, and
- *Schedule Generation*: Use active vs. non-delay procedure for schedule generation

The above implementation choices result in eight combinations, for each combination we tested 50 10x10 instances (i.e., 5 base problems each repeated for 10 replications for processing time variation). The results are summarized in Table 3.4. We will first make a few observations on the implementation differences before providing more detailed analysis in subsequent sections.

Results:

- Contrary to the ATC benchmark in Table 3.3 active schedule generation works significantly better under the PFSL scheme.
- Among the active scheduling results, using weak Lagrangean ranking for dynamic dispatching appears to be better than using the ATC-ranking in almost all cases. Interestingly, the situation reverses among the non-delay scheduling results: ATC-ranking yields better tardiness in most cases.
- Comparing the results of different upper bound evaluation during the subgradient search (using scaled vs. unscaled processing times), we see that the results are quite similar and there is no clear dominance between either implementation.

3.4.4 Computational Results

We conducted a second set of experiments for larger size (20x15 and 30x10) test problems using unscaled processing times, weak Lagrangean ranking and active schedule generation. In the following we examine the conjectures stated in Section 3.2.2 using the 10x10, 20x15 and 30x10 computational results.

3.4. COMPUTATIONAL STUDY

Tests for Conjecture 2: The Value of Second Moment Information

The second conjecture stated earlier claims that preprocessing strategies that incorporate the second moment information of processing time variation are more robust than ones that use only expected values. We test this conjecture by comparing the 80% and 50% confidence levels (β) used to characterize the chance constraints. Recall that in the 80%-confidence case both the mean and variance of processing time variations are used in Lagrangean preprocessing, while in the 50% cases the variance information is ignored. As the results in Table 3.4 show, in 85.4% (or 41 out of 48) of the cases the 80%-confidence cases yield significantly lower (an average of 9.7%) weighted tardiness when compared to the 50% cases.

The conjecture also state that the value of the second moment information increases as the level of uncertainty increases. As can be computed from Table 3.4, in the high uncertainty ($V = 50\%$) cases, the algorithm benefits from the use of the variance information much more frequently then the low uncertainty ($V = 10\%$) cases (94% vs. 69%). However, the amount of improvement is roughly the same (10.8% vs. 10.2%).

Furthermore, the value of the second moment information is affected by the type of distribution. It yields a performance improvement of up to 29.6% (11.3% average) in the Uniform cases, and up to 19.3% (7.8% average) in the Normal cases.

The effect of using the second moment information is further depicted in Figures 3.2 and 3.3 where we plot the results from the implementation using weak ranking, unscaled processing times and active scheduling. As the graphs show, the 80% confidence level draws the lower envelope in both variation types and in all variation levels. Corresponding plots for large-size problems are shown in Figures 3.4 and 3.7. As the figures show, the results from 20x15 and 30x10 problems support the conjecture that the second moment information significantly improves scheduling robustness.

The figures also show that the performance improvement achieved by the second moment information (comparing the 80%- and the 50%-confidence curves) is more pronounced in Uniform distribution cases.

3.4. COMPUTATIONAL STUDY

Result: The above observations suggest that the second moment information add significant value to the *a priori* analysis used for preprocessing. Moreover, this information has a higher marginal value when the level of uncertainty is higher (as is the case when $V=30\%$, 50%), and when the processing time variation is Uniformly (rather than Normally) distributed.

Tests for Conjecture 3: The Value of *A Priori* Analysis

In the third conjecture, we state that the performance of dynamic dispatching such as ATC can be significantly improved by *a priori* analysis using stochastic information. To test this conjecture we compare the results from straight-forward ATC dispatching (Table 3.3) and ATC dispatching after Lagrangean preprocessing (i.e., ATC-ranking for dynamic dispatching in Table 3.4). As shown from the tables, if the Lagrangean preprocessing is implemented using the second moment information ($\beta=80\%$) and active schedule generation, it significantly outperforms straight-forward non-delay ATC dispatching under all levels of uncertainty in all test cases (an average of 34% improvement, ranging from 7% to 80%). Figures 3.2-3.7 show graphically the performance of this preprocessing scheme (with and without the variance information) as compared to the non-delay ATC heuristic. As the graphs show, the performance of Lagrangean Preprocessing clearly dominate that of ATC.

Conjecture 3 also states that the extent of the improvement varies depending on the level of uncertainty, problem instances and the underlying distribution. Interestingly, the problem size also plays a role here. For 10x10 problems, we observe that the performance difference over ATC is consistent across different level of uncertainty. While for 20x15 and 30x10 problems the performance difference is more significant at high uncertainty levels. On the other hand, the distribution type does not seem to have a significant effect to the extent of improvement.

Result: The above observations suggest that *a priori* analysis using stochastic information can significantly improve the performance of dynamic dispatching. The extent of the improvement varies by the problem size and the level of uncertainty.

3.4. COMPUTATIONAL STUDY

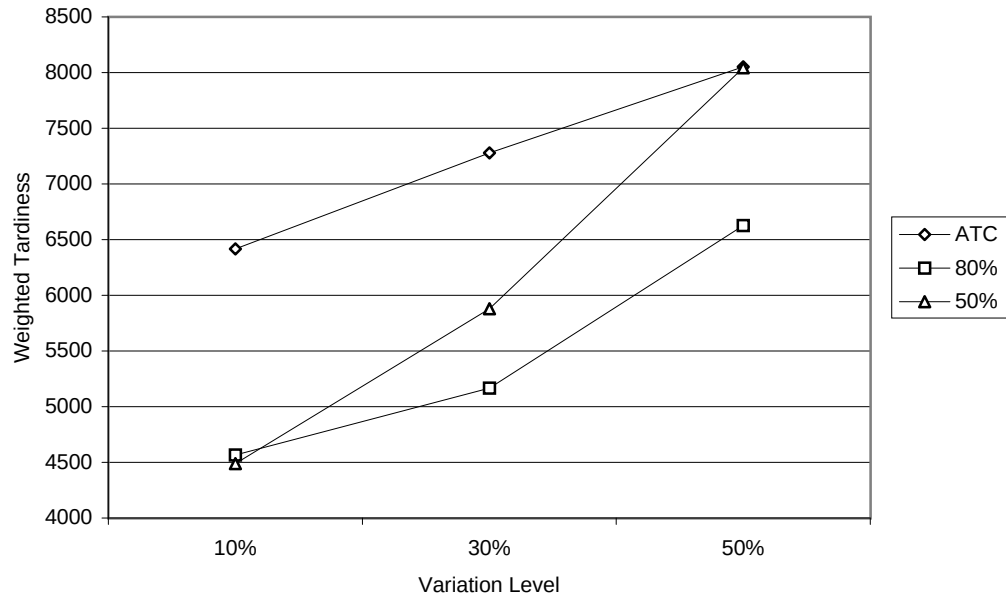


Figure 3.2: Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in *Uniform* processing time variation case (10x10 problems)

3.4. COMPUTATIONAL STUDY

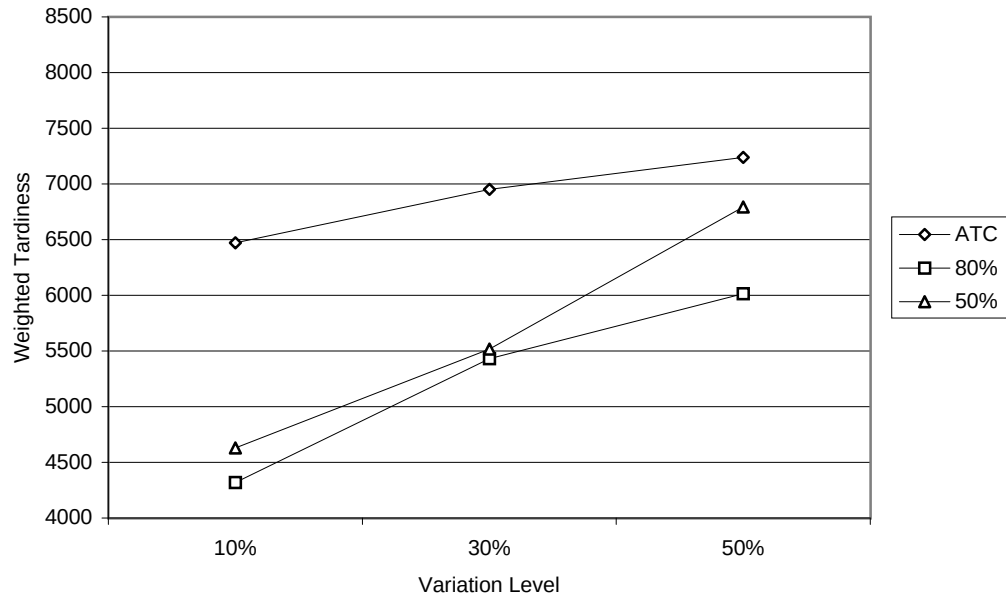


Figure 3.3: Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in *Normal* processing time variation case (10x10 problems)

3.4. COMPUTATIONAL STUDY

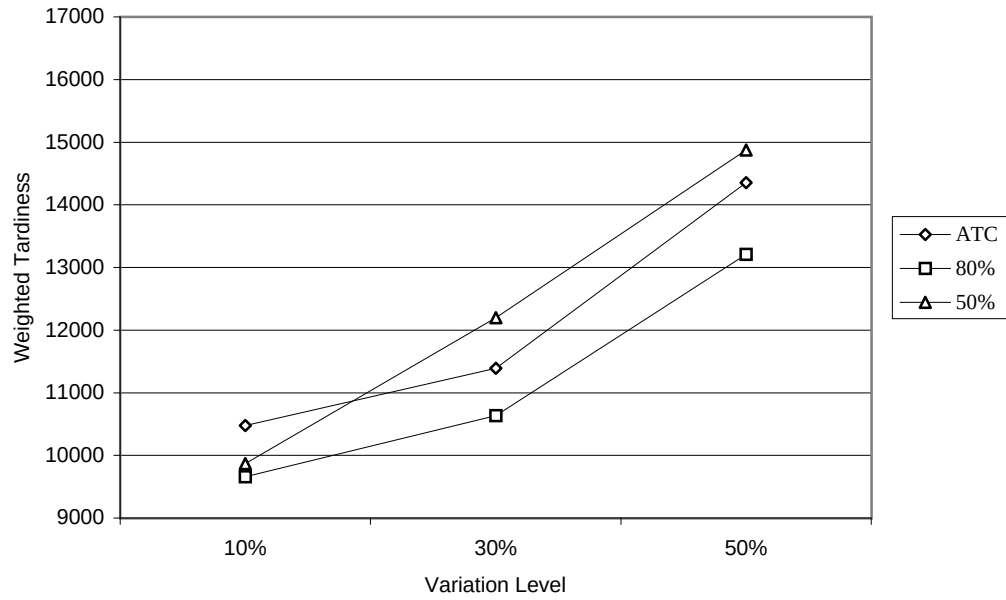


Figure 3.4: Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in *Uniform* processing time variation case (20x15 problems)

3.4. COMPUTATIONAL STUDY

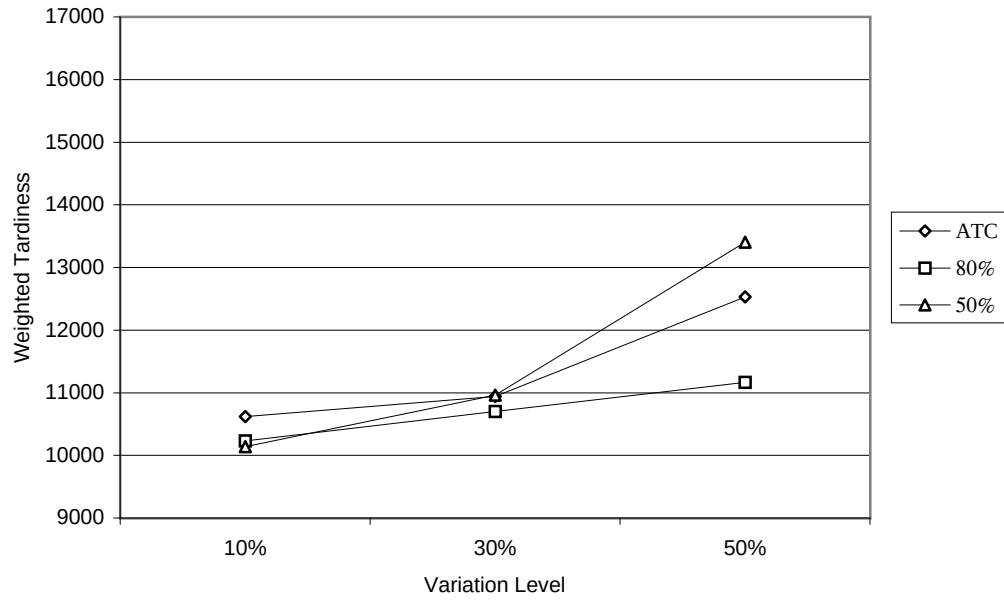


Figure 3.5: Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in *Normal* processing time variation case (20x15 problems)

3.4. COMPUTATIONAL STUDY

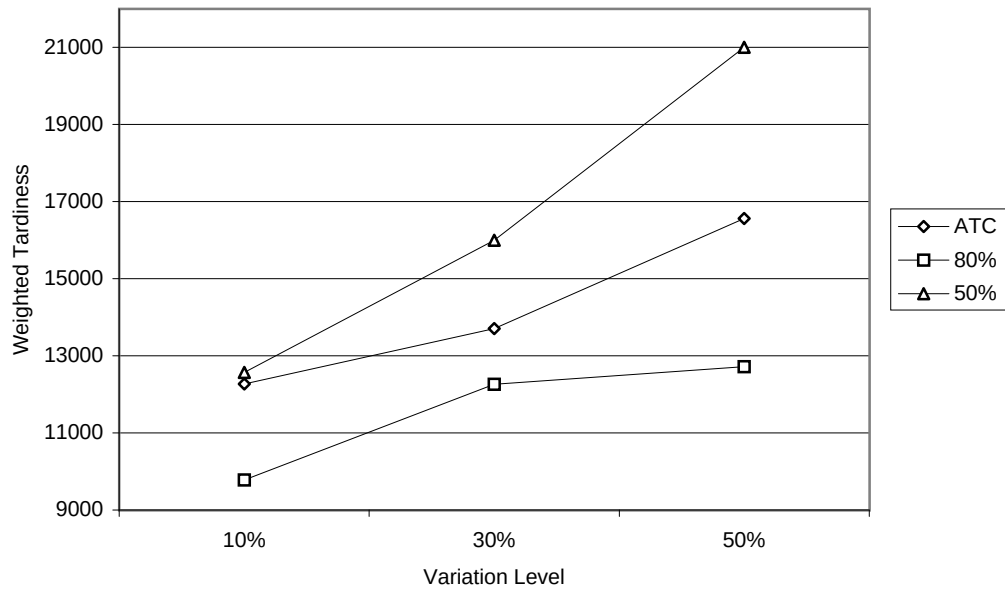


Figure 3.6: Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in *Uniform* processing time variation case (30x10 problems)

3.4. COMPUTATIONAL STUDY

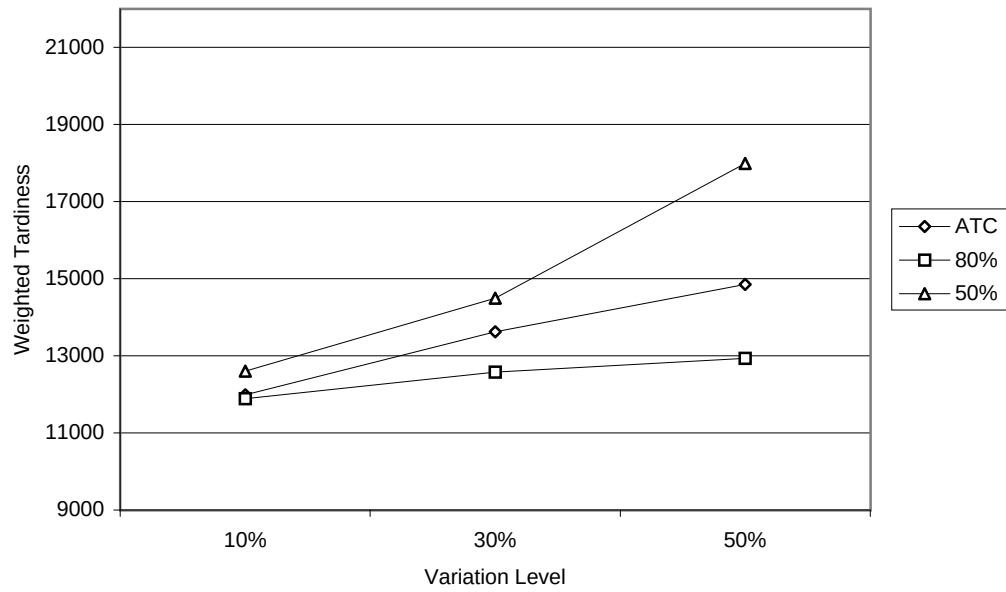


Figure 3.7: Average Weighted Tardiness versus Uncertainty Level for selected PFSL settings in *Normal* processing time variation case (30x10 problems)

3.4. COMPUTATIONAL STUDY

Tests for Conjecture 4: The Marginal Benefit of Stochastic Analysis

Conjecture 4 states that the marginal benefit of *a priori* stochastic information increases as the level of uncertainty increases, and its effect varies depending on the characteristics of its underlying statistical distributions. We will examine the first half of this conjecture using Figures 3.2-3.7. As the figures show, while the performance of ATC and 50%-confidence preprocessing without variance information (both methods ignore the stochastic information) deteriorate rapidly as the level of uncertainty increases, 80%-confidence preprocessing appears to be very robust to the changes in processing time uncertainty. This supports the conjecture that the stochastic analysis has an increasing importance as the level of uncertainty increases.

Now consider the second half of the conjecture. As we compare the slopes of the curves, we see that all methods are more sensitive to the uncertainty levels under Uniform distribution. As pointed out earlier, the plots also show that the performance improvement achieved by the second moment information (comparing the 80%- 50%-confidence cases) is more pronounced in the Uniform distribution cases. Recall that we have equalized the variance between the Uniform and the Normal distributions. If we compare the Uniform vs. Normal distribution results in Table 3.4, (at three corresponding levels of uncertainty), we find that 71% of the time (or 34 out of 48 cases) the preprocessing algorithm achieve a better performance under Normally distributed variations.

Result: The above observations suggest that *a priori* stochastic analysis indeed improve scheduling robustness. Compared to methods ignoring stochastic information, its benefit is more evident when (1) the level of uncertainty is high, and (2) when the variation is Uniformly distributed. However, comparing different statistical assumptions (Uniform vs. Normal distribution under equalized variance), Normal distribution appears to provide more valuable *a priori* information.

3.4. COMPUTATIONAL STUDY

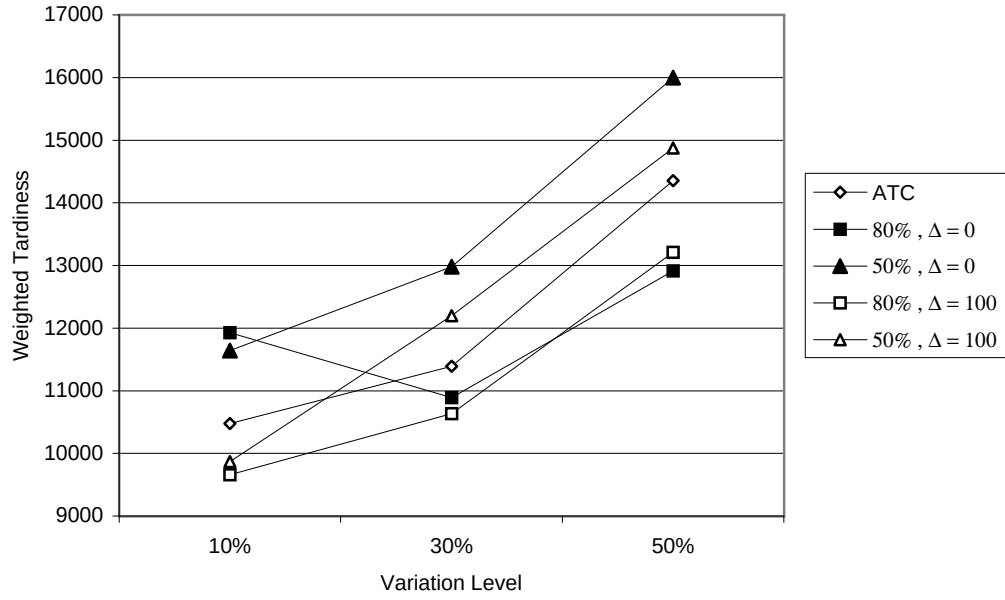


Figure 3.8: Effects of surrogate scheme ($\Delta = 100$) on PFSL algorithm. (Uniform processing time variation case, 20x15 problems)

3.4. COMPUTATIONAL STUDY

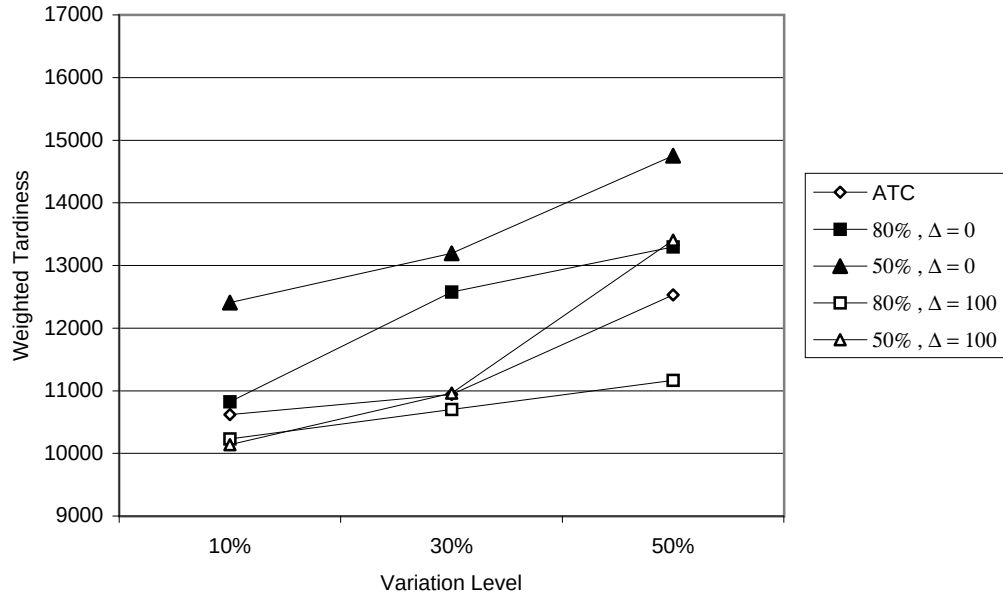


Figure 3.9: Effects of surrogate scheme ($\Delta = 100$) on PFSL algorithm. (Normal processing time variation case, 20x15 problems)

3.5. CONCLUSIONS

Remarks on Due Date Tightness

As the results presented in the previous sections show, problem instances do affect the performance of the PFSL scheme. When we first tested the 20x15 problem instances an interesting observation was made: during preprocessing the Lagrangean relaxation lower bound did not improve over 200 subgradient iterations, and the performance of PFSL was no better than that of a straight-forward ATC. Further investigation revealed that the LP-relaxation lower bound (which presents a theoretic upper bound for the Lagrangean relaxation lower bound in our model) is equal to 0 for almost all problems in this test set. That is, if we relax the binary constraints on the decision variables, we can process all the jobs on or before their due dates and obtain 0 weighted tardiness. Hence, the subgradient search algorithm does not have the needed information to improve the LR lower bound (the LB is always 0).

To address this issue we solve a surrogate Lagrangean relaxation problem during the preprocessing phase. The surrogate problem is created by subtracting a fixed value Δ from the original due dates for the LR problem and solving the revised problem using the subgradient search algorithm. At the end of the search we use the established Lagrangean ranking (both *full* and *weak ranking*) as before. When simulating the performance of this PFSL scheme, we use the original due dates. With this simple scheme, we have successfully overcome the “degeneracy” problem during the subgradient search. Figures 3.8 and 3.9 show the results of this set of experiments for 20x15 problems. As the graph depicts, the surrogate scheme significantly improve the performance of the preprocessing algorithm. All 20x15 results shown in earlier sections correspond to this improved version.

3.5 Conclusions

We have developed a preprocess-first-schedule-later scheme for job shop scheduling with the purpose of improving scheduling robustness. We first perform a Lagrangean relaxation of the machine capacity constraints which results in network-structured job

3.5. CONCLUSIONS

subproblems. Based on stochastic information about the jobs, we add stochastic processing time constraints to the job subproblems. The resulting chance-constrained stochastic program makes use of the first and second moment information on the processing time. This extra information comes at little computational expense since the subproblems still have the integrality property. Using a subgradient search algorithm, we preprocess each scheduling instance by solving the stochastic model. After preprocessing, a Lagrangean Ranking is established for the schedule. The actual schedule is generated dynamically using the Lagrangean ranking. Intensive computational testing supports many of the conjectures listed in Section 3.2.2. The main results are summarized at the end of each subsection in Section 3.4.

We demonstrated that both stochastic analysis and dynamic adaptation could significantly improve scheduling robustness. Making use of second moment stochastic information achieves significant improvement in a majority of the cases. More importantly, the benefits of stochastic analysis are more significant in larger size problems at a higher level of uncertainty. Not surprisingly, the marginal benefit of stochastic preprocessing is also affected by the assumed statistical distributions and problem instances.

Chapter 4

Proactive Equilibrium for Distributed Scheduling

4.1 Introduction

The purpose of this study is to analyze the theoretical requirements for distributed scheduling problems. We consider a distributed decision process where interdependent decision agents compete for resources while making decisions based on their local utilities, preferences and constraints. The main objective for distributed scheduling is to make globally sound decisions and to meet system-wide resource constraints while preserving the information privacy and local interests of decision agents. We consider the case where each decision agent represents the interests of a job that has multiple operations to be processed in a job shop. Each decision agent represents exactly one job, this job agent requests resources for each of its operations so as to optimize its own utility while satisfying its local constraints. We assume a price-directive mechanism for the coordination among agents. For a formal specification of this price-directive distributed scheduling system, see [46]. Several important issues arise when considering price-directive distributed scheduling. In the following, we briefly summarize some of these issues and their relationship to this research.

Solution Feasibility: A main issue in price-directive scheduling environments is that the collective requests made by the agents may not correspond to a feasible

4.1. INTRODUCTION

schedule. In fact, it represents a lower bound solution to the original scheduling problem with relaxed capacity constraints. In order for this infeasible schedule to converge to an optimal or a feasible schedule, iterative price adjustment needs to be followed by some feasibility restoration procedure. As shown in [46], a non-zero duality gap is to be expected for job shop scheduling problems under a typical subgradient mechanism. An augmented objective function can be used to eliminate the duality gap, but would not guarantee optimality. We propose an alternative approach to address this problem.

Assumptions on Information Availability: Another issue is the assumption on which type of information is available, and when, to the decision agents. The information assumption is important as it provides the basis not only for the design of the scheduling mechanism, but also the achievable efficiency of the mechanism. Scheduling information may be offered with different scope at different levels of specificity. For instance, when the scheduling procedure is used in a regular basis for resource allocation, say generating weekly schedules in a job shop, it is reasonable to assume that the agents have access to scheduling information accumulated over time. Since the patterns of resource loading tend to repeat themselves over time, this *a priori* information may help the scheduling procedure for quick convergence. A more conservative assumption would be that each scheduling task is unique and only the information available in the earlier stages of the current search can be used. Our study will focus on the latter assumption, while making use of the *a priori* information in the design of the scheduling mechanism.

Communication Overhead: Communication overhead is well known to be a significant issue in distributed systems (see, e.g., [58, 88]). In order for a distributed scheduling mechanism to be practical, it is important that most computing required for scheduling be done locally by the agents, and only minimal communication be required among the agents. This being the case, we may state the goal for distributed scheduling as “To generate a globally optimal schedule using locally autonomous decision agents that require minimal mutual communication.” The issue of communication efficiency is also one of the main focus of this study.

To address the above issues, we introduce in this study the notion of *proactive*

4.2. PROACTIVE EQUILIBRIUM

equilibrium. The basic idea is to make each agent consider an aggregate model of other agents in its utility maximization problem. The main elements of proactive equilibrium are given in the following.

4.2 Proactive Equilibrium

A basic motivation behind *proactive equilibrium* (PE) is that in traditional price-directive search methods or equilibrium-seeking tatonnement, each agent solves its own utility maximizing problem assuming that it is the only player in the system. Changes in the price vector are the only feedbacks an agent gets concerning the nature of the competition (i.e., what other agents need). In observing the search process it becomes obvious that each agent gradually changes its solutions over the iterations in response to the price changes. If the update scheme is constructed properly the solutions do eventually converge. This price-directive search paradigm can be viewed as a conservative process in which multiple agents negotiate toward an agreement. Two issues stated earlier are directly related to this price-update scheme: (1) communication overhead, which can be interpreted as the number of iterations needed to achieve convergence, and (2) solution feasibility, which corresponds to a non-zero duality gap between the converged (lower bound) solution and the optimal. *Proactive equilibrium* is a means of addressing these issues by incorporating a “model of others” in each agent’s subproblem. This “model of others” is used not only as a means of speeding up convergence (therefore reducing the communication overhead), but also as a tool for closing the duality gap. In the following, we first provide a generic discussion of the proactive equilibrium concept and then discuss the specifics of the equilibrium for distributed job shop scheduling.

The basic element of proactive equilibrium is that each agent, while solving its own utility optimization problem, incorporates an aggregate model of other agents. However, instead of explicitly considering other agents’ actions, this model represents an agent’s informed perception about other agents’ collective actions, and how these actions may limit its own actions. The agent updates this model using up-to-date

4.2. PROACTIVE EQUILIBRIUM

information provided by the coordinating mechanism (the center), and direct interactions with other agents. Each agent makes/alters its decisions based on this model of others “proactively.” Corresponding to some of the familiar concepts in noncooperative games, a basic distinction of PE to *Nash Equilibrium* and *rationalizability* (as defined by [59]) is that, in Nash Equilibrium each agent’s decision is a best response to the strategies *actually played* by its opponents, and *rationalizability* requires that an agent’s strategy be a best response to some justifiable conjecture about what its opponents *will be playing*, while proactive equilibrium only requires conjectures on the *collective impact* of its opponents to its own decision, regardless of their individual actions. Another interpretation of PE is that it is one of the Nash Equilibria for a “modified game” where each agent considers “perfect” or actual model of others which are incorporated in terms of constraints. Realizing that these constraints can be viewed as different prices for agents, PE is a Nash Equilibrium with a *discriminatory* (or nonlinear) pricing. When each agent’s *model of others* coincides with the realized actions of other agents, proactive equilibrium is reached (Although used in a different setting, PE can be viewed as a generalization of “conjectural variation” where firms competing in a market consider the effects of other firms on the prices and quantities of goods).

As expected, incorporating the *model of others* affects the utility function, denoted by U_i , and/or the solution space, denoted by $\mathcal{X}_i$, of the agent subproblem under consideration. Hence, the solution the agent obtains solving a local problem with a model of others will differ from that of the problem without such a model.

More formally, let $\hat{\omega}_i(\phi)$ be the predicted aggregate model of all agents but i (model of others) as perceived by agent i in state ϕ . State ϕ corresponds to the state of the system as defined by currently available public information accessible by all agents. The actual aggregate model of all agents except i is represented as $\omega_i(\phi, x^*)$ where x^* represents the optimal actions of all agents with respect to state ϕ . Normally this corresponds to the part of a overall feasible solution that limits agent i ’s actions. We then define the *proactive equilibrium* as follows:

Definition 4.1 Proactive Equilibrium *Given a system state ϕ , predicted models of agents $\hat{\omega}_i(\phi)$ ($i = 1, \dots, N$), and their corresponding solutions x_i^* constitute a*

4.3. THE JOB SHOP SCHEDULING MODEL

proactive equilibrium, if, for each agent i ,

$$\hat{\omega}_i(\phi) = \omega_i(\phi, x^*)$$

where x_i^* is the maximizer of agent i 's utility function, $U_i(\phi, x_i, \hat{\omega}_i(\phi))$, subject to $x_i \in \mathcal{X}_i(\hat{\omega}_i(\phi))$.

An important feature of proactive equilibrium is the flexibility it provides in utilizing available information. While each agent constructs and then updates its *model of others* using both public information provided by the center, and its own insights, the sources of information base on the following: (1) historic information on past transactions, and (2) real-time information from current interactions (earlier iterations of the current tatonnement). The former can be used as an *initial* model of others, whereas the latter provides feedbacks that update the *current* model. The historic information is available *a priori* and it accumulates over time. The *model of others* become more accurate over time since it provides a *long-term* memory about the system's past interactions. For this to be viable in a practical setting, the center may keep a database that keeps track of critical statistics of past scheduling activities, e.g., history of resource conflicts among agents, price history, resource loading profiles, etc.

To summarize, the basic elements of proactive equilibrium include (1) a precise definition for the *model of others*, $\hat{\omega}_i(\phi)$, (2) the agent's utility function including the model of others, $U_i(\phi, x_i, \hat{\omega}_i(\phi))$, and (3) a domain defining the agent's possible actions subject to the system state and the model of others, $\mathcal{X}_i(\hat{\omega}_i(\phi))$.

4.3 The Job Shop Scheduling Model

To further illustrate the notion of proactive equilibrium and its corresponding algorithmic structures in resource scheduling, we focus our study on the classical job shop scheduling model. In the classical job-shop scheduling model a set of jobs is to be completed on multiple machines. Each *job* consists of a series of *operations* that

4.3. THE JOB SHOP SCHEDULING MODEL

Table 4.1: Notation

i	Job index, $i = 1, \dots, N$ where N is number of jobs
j	Operation index, $j = 1, \dots, n_i$ where n_i is number of operations of job i
k	Machine index, $k = 1, \dots, M$ where M is number of machines
t	Time index, $t = 1, \dots, T$ where T is number of time periods in planning horizon
W_i	Weight or tardiness penalty of job i
d_i	Due date of job i
C_i	Completion time of job i
T_i	Tardiness of job i , $\max\{0, C_i - d_i\}$
m_{ij}	Machine required for operation j of job i
o_{ik}	Operation that job i requires from machine k , $m_{ij} = k \iff o_{ik} = j$
p_{ij}	Processing time of operation j of job i
C_{ij}	Completion time of operation j of job i , $C_{ij} = T - \sum_t X_{ijt} + p_{ij}$
S_{ij}	Start time of operation j of job i , $S_{ij} = T - \sum_t X_{ijt} + 1$
$v(P)$	Value of optimal solution of problem P

represent the production steps of the job. Due to processing requirements the operations of a job are subject to serial *precedence constraints*. Each production step (an operation) requires a specific *machine* for certain amount of time called *processing time*, and once the processing of an operation starts it can not be preempted by any other operations. The collection of operation *processing times* and their corresponding machine requirements represent the *job routing*. Each machine can only process one operation at a time. This requirement defines the *machine capacity constraints*. Each job has a *due-date* and a *weight*, and if the job is completed after this date the linear tardiness costs incur with associated weights. Hence, the objective is to find a feasible schedule which minimizes the total weighted tardiness.

We first state a discrete time-bucket integer programming formulation of the job-shop scheduling problem due to Pritsker et al. [73]. The notation is summarized in

4.3. THE JOB SHOP SCHEDULING MODEL

Table 4.1. The decision variable of this model (X_{ijt}) is defined as follows:

$$X_{ijt} = \begin{cases} 1 & \text{if operation } j \text{ of job } i \text{ has started by time } t, \\ 0 & \text{otherwise.} \end{cases} \quad (4.1)$$

The JSP can be formulated as follows (We set the variables and parameters with out-of-range indices to 0):

(JSP)

$$\min \sum_i W_i \left[\sum_{t > d_i - p_{i,n_i}} (1 - X_{i,n_i,t}) \right] \quad (4.2)$$

$$\text{s.t. } X_{i,j,t+1} \geq X_{i,j,t}, \quad \forall i, j, t < T \quad (4.3)$$

$$X_{i,j,t} \leq X_{i,j-1,t-p_{i,j-1}}, \quad \forall i, j > 1, t \quad (4.4)$$

$$\sum_{i,j:m_{ij}=k} (X_{i,j,t} - X_{i,j,t-p_{ij}}) \leq 1, \quad \forall k, t \quad (4.5)$$

$$X_{i,j,t} \in \{0, 1\}, \quad \forall i, j, t \quad (4.6)$$

The first set of constraints (4.3) makes sure that once an operation is started, it remains so in all subsequent time periods. This is due to the definition of X_{ijt} and the non-preemption requirement. Constraints (4.4) state that an operation cannot start until all its predecessors are completed. Finally, the machine capacity constraints (4.5) state that at most one job can be processed on a particular machine in a given time period. The objective function (4.2) is the total weighted tardiness derived from the fact that a cost of W_i is incurred for each time period after d_i during which job i has not been completed.

We rewrite the objective function as follows:

$$\sum_i W_i \left[\sum_{t > d_i - p_{i,n_i}} (1 - X_{i,n_i,t}) \right] = \sum_i W_i (T - d_i + p_{i,n_i}) + \sum_i \sum_j \sum_t -w_{ijt} X_{ijt} \quad (4.7)$$

where

$$w_{ijt} = \begin{cases} W_i & \text{if } j = n_i \text{ and } t > d_i, \\ 0 & \text{otherwise.} \end{cases} \quad (4.8)$$

4.3. THE JOB SHOP SCHEDULING MODEL

Since the first term in the objective function (4.7) is a constant we represent it as $A = \sum_i A_i$ where $A_i = W_i(T - d_i + p_{i,n_i})$.

The job level subproblem can be formed by dualizing the machine capacity constraints (4.5) of (JSP) with nonnegative Lagrangean multipliers λ_{kt} , $k = 1, \dots, M$, $t = 1, \dots, T$. Then, we obtain the following Lagrangean Relaxation problem:

$(LR(\lambda))$

$$\min \sum_i A_i + \sum_i \sum_j \sum_t (\lambda_{m_{ij},t} - \lambda_{m_{ij},t+p_{ij}} - w_{ijt}) X_{ijt} - \sum_k \sum_t \lambda_{kt} \quad (4.9)$$

subject to (4.3), (4.4), and (4.6)

To form independent job-level subproblems we decompose $LR(\lambda)$ into $(LR_i(\lambda))$ as follows:

$$v(LR(\lambda)) = \sum_i v(LR_i(\lambda)) - \sum_k \sum_t \lambda_{kt} \quad (4.10)$$

where

$$v(LR_i(\lambda)) = \min A_i + \sum_j \sum_t (\lambda_{m_{ij},t} - \lambda_{m_{ij},t+p_{ij}} - w_{ijt}) X_{ijt} \quad (4.11)$$

$$\text{s.t. } X_{i,j,t+1} \geq X_{i,j,t} \quad \forall j, t < T \quad (4.12)$$

$$X_{i,j,t} \leq X_{i,j-1,t-p_{i,j-1}} \quad \forall j > 1, t \quad (4.13)$$

$$X_{i,j,t} \in \{0, 1\} \quad \forall j, t \quad (4.14)$$

In the remainder of this study, we use this simple job level subproblem $(LR_i(\lambda))$ as a basis to define the job agent subproblem. In this context, we can interpret the value of a Lagrange multiplier λ_{kt} as the price associated with the use of a particular machine-time slot charged to the job agents in a price tatonnement. A convenient property is that each job subproblem is the dual of a specially structured maximum flow network problem [77]. Hence, each job-level subproblem can be solved using efficient network flow algorithms.

4.4 Proactive Equilibrium for JSP

Given the above setting, we will now define the main components for a proactive equilibrium in JSP.

Model of Others, $\hat{\omega}_i(\phi)$: The model of others addresses three fundamental issues of a proactive equilibrium in JSP: communication overhead, solution feasibility, and information utilization. Stated simply, the goal is to construct a model of others in the context of a price tatonnement such that (1) a majority of the computation is *local* to the job agent subproblem, (2) a *feasible* solution (or a set of solutions) results at the end of the tatonnement, and (3) the *information* accumulated during the tatonnement is utilized to improve efficiency. Having this in mind, we construct a concise model of others for JSP as *inserted constraints to the job agent subproblems*. These constraints force the operations of job i to start on or after certain time slots:

$$S_{ij} \geq \tau_{ij}(\phi), \quad j = 1, \dots, n_i \quad (4.15)$$

Note that the right hand side of these constraints, $\tau_{ij}(\phi)$, are not scalar values, but state-dependent parameters updated over the price tatonnement as the system state ϕ changes. The system state ϕ is defined as the public information available to all agents and $\phi \equiv (\lambda, \hat{\tau})$, where λ is the current resource pricing and $\hat{\tau}$ represent the last updated parameter value. Thus, we define the model of others in JSP:

$$\hat{\omega}_i(\phi) \equiv \{(\tau_{ij}(\phi)) : j = 1, \dots, n_i\}$$

In Section 4.5 we will examine three different types of constraints as the model of others.

Decision Domain, $\mathcal{X}_i(\hat{\omega}_i(\phi))$: To each job agent, the decision domain is defined by the current system state ϕ and the limits imposed by the model of others. Hence,

$$X_i \in \mathcal{X}_i(\hat{\omega}_i(\phi))$$

4.4. PROACTIVE EQUILIBRIUM FOR JSP

where $\mathcal{X}_i(\hat{\omega}_i(\phi))$ is the revised feasible space of job agent i defined by the original job level constraints and the new *start time constraints* as constructed in (4.15).

Job Agent Subproblem: The utility function $U_i(\phi, \hat{\omega}_i(\phi), X_i)$ for job agent i is defined as the job-level objective in the problem $(LR_i(\lambda))$. To simplify the following discussion, we state the job agent subproblem as a minimization problem rather than the more conventional utility maximization problem. Thus, the job agent subproblem is defined as finding an optimal job agent schedule $X_i^*(\phi)$ that minimizes function $U_i(\phi, \hat{\omega}_i(\phi), X_i)$ subject to $X_i \in \mathcal{X}_i(\hat{\omega}_i(\phi))$.

Actual Model of All Agents: The actual model of all agents must be computed by the center using a feasibility restoration routine. In this case, the actual model of all agents $\omega_i(\phi, X^*(\phi))$ refers to the starting times of job i in a feasible solution restored from the collection of job agent subproblems with respect to given state ϕ .

We can now define proactive equilibrium for JSP as follows:

Definition 4.2 (Proactive Equilibrium for JSP, PEJSP) *Given a system state ϕ , predicted models of other agents, $\hat{\omega}_i(\phi) \equiv \{(\tau_{ij}(\phi)) : j = 1, \dots, n_i\}, i = 1, \dots, N$, and their solutions $X_i^*(\phi), i = 1, \dots, N$ constitute a proactive equilibrium, if, for each job agent i ,*

$$\hat{\omega}_i(\phi) = \omega_i(\phi, X^*(\phi))$$

where $X_i^*(\phi)$ is the utility optimizer of $U_i(\phi, \hat{\omega}_i(\phi), X_i)$ subject to $X_i \in \mathcal{X}_i(\hat{\omega}_i(\phi))$.

Note that for JSP, proactive equilibrium implies both feasibility and global optimality (as defined by the original JSP model). The existence of the equilibrium and the tatonnement required to achieve the equilibrium will be described in the following sections. Note that the equilibrium point may not correspond to the global optimal defined by the original JSP if we allow the jobs to make use of private information and further restrict themselves in domain $\mathcal{X}_i(\hat{\omega}_i(\phi))$. The same could happen to the constraints computed as the model of others. Thus, if optimality is of concern, a primary issue in PEJSP is to ensure that added constraints in the agent subproblems do not cut off the optimal from the solution domain. In many practical situations, single point optimality may not be of primary concern, then PEJSP can be reduced to a feasibility seeking framework given specified job agent utilities and constraints.

4.5. MODEL OF OTHERS

In this study, we focus our attention on the former case.

4.5 Model of Others: Inserting Constraints to the Job Agent Subproblem

We now consider the mechanics involved in constructing the model of others: inserting and updating constraints to the job agent subproblem. For this analysis, we denote a generic problem by P . Denote $v(P)$ as the value of the optimal solution for P , and D a set of constraints that involve the decision variables in P . $P(D)$ denotes a more restricted problem with the same objective and constraints as in P and additional constraints D . If problem P has any other parameters, say a vector λ , then the restricted version is denoted by $P(\lambda, D)$. We first show that if we insert constraints D that are already satisfied by an optimal solution to problem P , we are still able to obtain the same optimal value of the original problem.

Proposition 4.1 *For a given set of constraints D , if there exists an optimal solution for JSP that satisfies all the constraints in D , then $v(JSP(D)) = v(JSP)$.*

Therefore, if we can insert constraints that do not eliminate an optimal schedule of the original problem JSP , we can still achieve the same optimal value. The next proposition shows that if we insert these constraints to the Lagrangean dual problem $LRD(D)$, then it still provides a lower bound for the original problem.

Proposition 4.2 *For a given set of constraints D , if there is an optimal solution for JSP that satisfies all the constraints in D , then $v(LRD(D)) \leq v(JSP)$.*

Since the value $v(LRD(D))$ is the maximum attainable Lagrangean lower bound, the proposition is still true if we replace $v(LRD(D))$ with $v(LR(\lambda, D))$.

We now demonstrate that a proactive equilibrium always exists in JSP. We use the notation $P(\sigma)$ to denote a more restricted version of P with additional precedence constraints for the operations ranked by ranking set σ . If σ is singleton, say $\sigma = \{((i, j), (i', j'))\}$, then we use the notation $P((i, j) \rightarrow^k (i', j'))$ or shortly

4.5. MODEL OF OTHERS

$P(i \rightarrow^k i')$. We will call the precedence constraints that are implied by a ranking set as *ranking constraints*. In this case, ranking set σ can alternatively be viewed as a collection of disjunctive arcs that are fixed in the direction that is suggested by the associated ranking. Note that $JSP(\sigma)$ represents a preprocessed scheduling instance with partially established ordering of operations. We will consider only *consistent* ranking sets where the established ordering does not cause any cycles in the preprocessed disjunctive graph. Given the above setting, we can state the existence of PE-JSP as follows.

Proposition 4.3 *If an optimal solution exists for the JSP, then an proactive equilibrium also exists for the JSP where the model of others are defined by a ranking set $\sigma = \sigma^*$ derived from the optimal sequence, i.e., $v(LRD(\sigma)) = v(LR(0, \sigma)) = v(JSP)$.*

While showing the existence of PE-JSP is straightforward, constructing the mechanism to achieve proactive equilibrium is much more difficult. The key for such a mechanism is the construction of the *model of others*. In the subsequent sections we consider three types of constraints to be inserted to the Job Agent Subproblem as the model of others: (1) *Rank-based precedence constraints*, (2) *1 out of K constraints*, and (3) *CEST constraints*.

4.5.1 Rank-based Precedence Constraints

We first summarize the notion of Lagrangean Schedule and Lagrangean Ranking. As stated in our previous studies [46, 47], the solution of the LR problem (collections of job level solutions), not only gives a lower bound for the original problem but also provides valuable ranking information which can be used for constructing feasible schedules. A *Lagrangean Schedule* is a capacity-infeasible shop schedule constructed by combining all solutions from (capacity relaxed) job-level subproblems. *Lagrangean Ranking* is defined accordingly as follows:

Definition 4.3 (Lagrangean Ranking) *Consider three specific conditions between operation pairs (i, j) and (i', j') in the Lagrangean Schedule as follows:*

- (1) (i, j) and (i', j') require the same machine, i.e., $m_{ij} = m_{i'j'}$,

4.5. MODEL OF OTHERS

- (2) operation (i, j) starts earlier than (i', j') in the Lagrangean Schedule, i.e., $S_{ij} \leq S_{i'j'}$
- (3) there is no resource conflict between (i, j) and (i', j') in the Lagrangean Schedule, i.e., $(X_{i,j,t}^*(\lambda) - X_{i,j,t-p_{ij}}^*(\lambda)) + (X_{i',j',t}^*(\lambda) - X_{i',j',t-p_{i'j'}}^*(\lambda)) \leq 1, t = 1, \dots, T$.

If an operation pair (i, j) and (i', j') satisfies conditions (1)-(3) then we say that they are strongly ranked, and operation (i, j) has a higher priority than (i', j') in the Lagrangean ranking. For operation pairs which satisfy conditions (1) and (2) but not (3), we say that they are weakly ranked. All the other operation pairs are unranked.

It is clear from the definition that only operation pairs with the same machine requirement with no resource conflict in the *Lagrangean Schedule* are *strongly ranked*. All the others that are on the same machine are weakly ranked. That is, every operation pair $((i, j), (i', j'))$ with $m_{ij} = m_{i'j'}$ is either strongly ranked or weakly ranked.

For a given vector λ , suppose we first solve the regular (original) job subproblems $LR_i(\lambda)$, compute $X^*(\lambda)$ and the corresponding Lagrangean schedule, and then find the associated strong ranking, denoted by $\sigma(\lambda)$. We can actually add all the precedence constraints associated with the strong ranking to the relaxed problem, but the danger is the possibility of cutting off the optimal solution of the original problem. Consequently, the calculated optimal value of the restricted relaxed problem may no longer be a lower bound for the original problem, and any price directive search using the upper bound information may get stuck prematurely at a suboptimal solution.

The next proposition shows that if we insert hard precedence constraints that are already suggested by any subset of the current strong ranking set, then the current solution of the revised Lagrangean relaxed problem for a given multiplier vector will not change.

Proposition 4.4 *Let $\sigma(\lambda)$ denote the strong ranking set found from the Lagrangean solution of $LR(\lambda)$ for a given λ . Then, $v(LR(\lambda, \sigma')) = v(LR(\lambda, \emptyset)) = v(LR(\lambda))$ for any $\sigma' \subseteq \sigma(\lambda)$.*

4.5. MODEL OF OTHERS

Based on this property, we can conclude that the solution of the new problem is still a lower bound for a given price vector λ . However, we cannot guarantee this in general, especially for a different price vector, since we may have $v(LRD(\sigma')) = v(LR(\lambda^*, \sigma')) \geq v(JSP)$ for some $\sigma' \subseteq \sigma(\lambda)$. Intuitively, the possibility for this is smaller for smaller fixed ranking sets σ' extracted from $\sigma(\lambda)$. For example, if σ' is singleton, say $i \rightarrow^k i'$ in $\sigma(\lambda)$, then it is less likely to have $v(LRD(i \rightarrow^k i')) = v(LR(\lambda^*, i \rightarrow^k i')) \geq v(JSP)$ although $v(LR(\lambda, i \rightarrow^k i')) = v(LR(\lambda, \emptyset))$ for any given λ .

We will now consider inserting precedence constraints that are opposite to the directions suggested by the members of a strong ranking. Let σ^c be the set of rankings (and its associated precedence constraints) whose every member is a ranking that is in reverse direction of an element in σ . That is,

$$((i, j), (i', j')) \in \sigma \iff ((i', j'), (i, j)) \in \sigma^c$$

or more concisely, σ suggests $i \rightarrow^k i'$ if and only if σ^c suggests $i' \rightarrow^k i$. We will call σ^c the *negation* or *complement* ranking set of σ . An important result of this discussion is summarized in the following theorem:

Theorem 4.1 *Let $\sigma(\lambda)$ denote the strong ranking set derived from the solution of $LR(\lambda)$. If $v(LR(\lambda, s')) > UB^*$ for any $s' \subseteq \sigma^c(\lambda)$, then there is a subset s of $(s')^c$, where at least one ranking constraint must hold in an optimal solution.*

Proof: $v(LR(\lambda, s'))$ is a lower bound for the optimal solution of $JSP(s')$. If $v(LR(\lambda, s')) > UB^* \geq v(JSP)$, then $v(JSP(s')) > v(JSP)$. Then there is a subset of constraints in s' , denoted by s'' , which satisfy the following conditions: (1) $v(LR(\lambda, s'')) > UB^*$, and (2) $v(LR(\lambda, s'' - \{i \rightarrow^k i'\})) \leq UB^*$, for every $i \rightarrow^k i'$ in s'' . In this case, we know that $v(JSP(s'')) > v(JSP)$ and thus any optimal solution of JSP cannot have all the ranking constraints in s'' satisfied, otherwise its lower bound would be greater than our current best upper bound. This means at least one opposite direction should be satisfied, i.e., one constraint in ranking set $s = (s'')^c$, which is a already subset of the strong ranking set, must hold in an optimal solution of JSP . $\diamond$

4.5. MODEL OF OTHERS

Since $s' \subseteq \sigma^c$, $s \subseteq (s')^c \subseteq (\sigma^c(\lambda))^c = \sigma(\lambda)$. Hence, suppose we can find subsets of the negation ranking sets which satisfy the condition in Theorem 1, we know that one member from each corresponding strong ranking set must be satisfied at optimality. We call such subsets *valuable strong ranking subsets*, or shortly *valuable sets*. The above property allows us to convert strong rankings in a valuable set to “1 out of K ” constraints to job-level problems (where K is the number of ranking constraints in the valuable set).

Note that the smaller the valuable subsets, the stronger the result of the theorem. Suppose $s' = \sigma^c$ and $v(LR(\lambda, \sigma^c)) > UB^*$. Then we can safely state that one of the rankings in the whole strong ranking set $\sigma(\lambda)$ must hold. Hence, an obvious valuable set is the strong ranking set itself. However, this is not as strong as a condition that states one of the rankings in a smaller valuable subset should hold in an optimal solution. Then, the goal is to find *minimal* valuable subsets of the strong rankings that do not contain other valuable subsets. One seemingly unrelated tool to find such minimal valuable subsets in the strong ranking set is the infeasibility detection algorithms. These algorithms try to find irreducibly inconsistent subsystems (IISs) in an infeasible mathematical model. They provide efficient mechanisms to isolate IIS or minimal constraint subsets with the property that removing any constraint from the set results in a feasible model [15, 16]. In the following we will show that an IIS algorithm can be used to identify “minimal” subsets of the ranking constraints in s' (which is in σ^c).

Denote $LRO(\lambda, s')$ a new problem related to $LR(\lambda, s')$. In this problem, we have the same objective and the same set of constraints as in $LR(\lambda, s')$, but we insert an additional constraint stating that the objective value $v(LR(\lambda, s'))$ should be less than or equal to a best known upper bound UB^* . This means that a problem instance would be infeasible if $v(LR(\lambda, s')) > UB^*$. Thus, an IIS detection algorithm can be run to find a minimal IIS which would consists of some original constraints from $LR(\lambda, s')$ (which we know are feasible) as well as some ranking constraints in s' (which may or may not be valid in an optimal solution). Alternatively, since we know the added ranking constraints are the potential cause of this infeasibility, we may focus the IIS search on the ranking constraints solely. In this way, we can use an IIS detection

4.5. MODEL OF OTHERS

routine to isolate minimal “bad” ranking subsets in s' that collectively cause the violation $v(LR(\lambda, s')) > UB^*$. From this point of view, simple IIS detection algorithms such as *deletion filter* or *additive algorithm* are good candidates [15, 16]. These algorithms systematically delete or add individual constraints until the remaining model becomes feasible/infeasible. In this case, we can start with a core set of constraints (non-preemption, operation precedence constraints, the newly added objective function constraint), and add the ranking constraints in certain order. Whenever we find it infeasible, we constructed an IIS. Different ordering of ranking constraints should yield different IISs.

Each such IIS set found through an IIS algorithm consists of all the core constraints and a subset of s' , denoted by s'' . Using the property of IIS, we know that if we inserted the constraints in s'' the objective function value would be greater than UB^* , while no subset of s'' has this property. One obvious implication of this is that not all ranking constraints in s'' can be satisfied at the same time in an optimal solution (since its lower bound is greater than the best known upper bound), or according to Theorem 1, at least one of the ranking constraints in $(s'')^c$ should be satisfied at optimality. Note that by taking the negation of s'' , we are back in the strong ranking “space”, i.e., $s \equiv (s'')^c \subseteq \sigma$. Thus, we can safely insert a “1 out of K ” constraint that forces one of K ranking constraints in the (minimal) valuable subset $s = (s'')^c$ to be satisfied to the original problem $LR(\lambda)$. We now state the following proposition.

Proposition 4.5 *All valuable subsets obtained from a strong ranking using an IIS algorithm are minimal.*

The proposition follows from the minimality property of IISs, and the correspondence between the IISs and the valuable subsets. Using this proposition, we can conclude that none of the constructed valuable subsets can contain the ranking constraints from another valuable subset. This in turn implies that none of the 1 out of K constraints obtained from the above IIS scheme is redundant. However, they may share some common ranking constraints; they need not be disjoint. Thus, in theory, the valuable subsets found using an IIS algorithm generates the “strongest” possible 1 out of K constraints that could be constructed.

4.5. MODEL OF OTHERS

Note that we can efficiently isolate multiple IISs using the IIS algorithm. In fact, the more and the smaller IISs (valuable sets) are, the better and tighter (closer to optimal) the job level problem become. In light of this, we may direct the IIS algorithm to focus on finding smaller (but more) IISs. Specifically, we may modify the IIS detection algorithm by specifying a threshold value for the cardinality of s'' as a means of limiting the size of the valuable subsets, K . We will denote the new relaxed problem by $LR(\lambda, I_{\sigma(\lambda)})$, where $I_{\sigma(\lambda)}$ denotes the set of inserted 1 out of K constraints along with the ranking constraints involved in these 1 out of K constraints.

We will give a simple example in the following. Suppose a strong ranking set for given λ suggests the following: $\sigma(\lambda) = \{1 \rightarrow^2 2, 2 \rightarrow^1 3, 1 \rightarrow^1 3\}$. Further, assume we find an IIS for $LRO(\lambda, \sigma^c(\lambda))$ consisting of non-preemption, operation precedence, objective constraints together with s'' , where $s'' = \{2 \rightarrow^2 1, 3 \rightarrow^1 2\}$, i.e., $v(LR(\lambda, s'')) > UB^*$. Then we can conclude that an optimal schedule cannot have both $2 \rightarrow^2 1$ and $3 \rightarrow^1 2$. Alternatively, either $1 \rightarrow^2 2$ or $2 \rightarrow^1 3$ in an optimal solution. In this case, $n = 2$, and the corresponding 1 out of K constraint is as follows:

$$Y_{122} + Y_{231} = 1,$$

where $Y_{i,i',k} = 1$ if job i is scheduled before i' on machine k , 0 otherwise. In this case, the related ranking constraints can be formulated in the following way:

$$S_{ij} \geq S_{i',j'} + p_{i',j'} - \mathcal{M}Y_{i,i',k}$$

where $\mathcal{M}$ is a large constant number. Note that the introduction of variables Y is similar to the variables used for dealing with regular disjunctive constraints. However, while the regular disjunctive relations are binary and they consider each pair of jobs on a specific machine, our 1 out of K constraints can bring together different job combinations (possibly with less or more than 2 jobs) on different machines.

4.5. MODEL OF OTHERS

4.5.2 Special Case: 1 out 1 constraints

There is a special case of Theorem 4.1, in which s'' is singleton. We present this as a corollary to the theorem.

Corollary 4.1 *If $\sigma(\lambda)$ suggests $i \rightarrow^k i'$, and $v(LR(\lambda, i' \rightarrow^k i)) > UB^*$, then $i \rightarrow^k i'$ is in an optimal ranking, i.e., $((i, j), (i', j')) \in \sigma^*$ where σ^* is an optimal ranking of the original problem JSP.*

This result shows that if we are lucky enough to find an IIS with only one ranking constraint, $s'' = \{i' \rightarrow^k i\}$, then we can conclude that the ranking $i' \rightarrow^k i$ cannot be in an optimal solution. In other words, i should be scheduled before i' on machine k , i.e., we can safely insert the constraint

$$S_{i',j'} \geq S_{ij} + p_{ij}$$

where $j = o_{ik}$ and $j' = o_{i',k}$. This basically corresponds to a 1 out of 1 constraint for the minimal valuable subset $s = \{i \rightarrow^k i'\}$. Since we know that no other minimal valuable subsets can include this ranking, we can exclude the reverse of it from further consideration in the search of remaining IISs.

In fact, when we add a hard ranking constraint between i and i' in the reverse direction ($i' \rightarrow^k i$) to the original LR problem (without any other constraints), we only need to solve a two-job LR problem to calculate $v(LR(\lambda, i' \rightarrow^k i))$. Note that the solutions of all other jobs are not affected from this constraint, since there is no other ranking constraints that tie them together. Thus,

$$v(LR(\lambda, i' \rightarrow^k i)) = \sum_{i'' \neq i, i'} v(LR_{i''}(\lambda, \emptyset)) + v(LR_{i,i'}(\lambda, i' \rightarrow^k i)) - \sum_k \sum_t \lambda_{kt} \quad (4.16)$$

where $LR_{i,i'}(\lambda, i' \rightarrow^k i)$ is defined as a problem with an objective as the summation of the two jobs' objectives, with all the constraints that come from their original job level subproblems, and additional precedence constraint that requires job i' to be processed before job i on machine k . Note that the first term is not affected by the reverse arc fixing, thus it is just a summation of the already computed job-level

4.5. MODEL OF OTHERS

subproblem objectives. However, if we consider the reverse ranking constraint on top of some fixed ones, denoted by σ , then the calculation on the right hand side of the formula (4.16) provides a lower bound on the true value of $v(LR(\lambda, \sigma \cup \{i' \rightarrow^k i\}))$, since the new ranking constraint might affect all the jobs' solutions, i.e., we may have to solve $LR(\lambda, \sigma \cup \{i' \rightarrow^k i\})$ as it is.

We note that Corollary 4.1 represents a result similar to the *immediate selection rule* developed for makespan-based job shop scheduling problem [13]. From this point of view, we can view Theorem 4.1 as a generalization of the immediate selection rule.

4.5.3 Properties of the Job Agent Subproblem with 1 out of K Constraints

We now know that the constraints in $LR(\lambda, I_{\sigma(\lambda)})$ not only do not exclude the optimal solution, but these added precedence constraints partially characterize the optimal solution(s). With a smaller feasible space (less number of solutions), the new 1 out of K -type constraints in $LR(\lambda, I_{\sigma(\lambda)})$ most certainly tightens the original LR lower bound. The next proposition states this result:

Proposition 4.6 *Let $\sigma(\lambda)$ be a strong ranking set that is constructed from the solution of $LR(\lambda, \emptyset)$. Then, $v(LR(\lambda, \emptyset)) \leq v(LR(\lambda, I_{\sigma(\lambda)}))$.*

If we optimize the Lagrangean dual of the new problem $LR(\lambda, I_{\sigma(\lambda)})$, denoted by $LRD(I_{\sigma(\lambda)})$, for optimal λ^* (which is most likely different from λ for which $\sigma(\lambda)$ is derived), we know that we can get a tighter lower bound than the original Lagrangean dual value. Moreover, we also know that this maximum attainable value of the lower bound for the revised problem is still a lower bound for the optimal solution of the original problem (Recall the propositions in Section 4.5). This means that the new formulation has a potential to narrow or ideally close the duality gap. This is stated in the following:

Proposition 4.7 *Let $\sigma(\lambda)$ be a strong ranking set that is constructed from the solution of $LR(\lambda, \emptyset)$. Then, $v(LRD) \leq v(LRD(I_{\sigma(\lambda)})) \leq v(JSP)$.*

4.5. MODEL OF OTHERS

Until now we have considered inserting additional constraints to a relaxed problem with original constraint set, i.e., we have just compared the new formulation $LR(\lambda, I_{\sigma(\lambda)})$ with the base case $LR(\lambda)$. However, note that in a price tatonnement such as a subgradient search we may construct a strong ranking set for each price vector λ s. Hence, we can find the minimal valuable subsets and the corresponding (potentially different) 1 out of K constraints over price-update iterations, while a newly identified set can be added on top of the set found in previous iterations. The properties outlined above hold under this condition in an accumulative fashion, i.e., we may keep inserting new constraints to job subproblems so long as these constraints preserve and further characterize the rankings in an optimal solution. By continuously identifying new valuable sets, we may tighten the lower bound over iterations and the solution always remains a relaxation of the original JSP. This result is summarized in the following proposition:

Proposition 4.8 *Let $\sigma(\lambda^1)$ and $\sigma(\lambda^2)$ be two strong ranking sets that are constructed for given vectors λ^1 and λ^2 , respectively. Define $I^1 = I_{\sigma(\lambda^1)}$ and $I^2 = I_{\sigma(\lambda^2)}$. Then,*

$$v(LRD) \leq \max\{v(LRD(I^1), v(LRD(I^2))\} \leq v(LRD(I^1 \cup I^2)) \leq v(JSP).$$

However, this additional improvement with $LR(\lambda, I_{\sigma(\lambda)})$ comes with a cost: The new Job Agent Subproblem is more difficult to solve since it involves 1 out of K constraints which introduces new binary variables Y to the problem. Without them, the Job Agent Subproblem is a collection of network flow problems which can be solved with linear algorithms. In the next section, we outline the use of CEST constraints which will prove useful in terms of computational complexity and implementability in the context of proactive equilibrium.

4.5.4 CEST Constraints

We now introduce “constrained earliest starting time”(CEST) constraints. Suppose we are given a preprocessed Lagrangean relaxation problem with a set of 1 out K constraints and established ranking constraints. We will denote the new problem by

4.5. MODEL OF OTHERS

$LR(\lambda, I)$, where I is now defined independently from a specific $\sigma(\lambda)$ to show that it is a collection of 1 out of K constraints and associated ranking constraints possibly found using many strong rankings from many multiplier vectors. We index each 1 out of K constraint with l and we assume that we have L such constraints, i.e., $l = 1, \dots, L$. We denote the set of ranking constraints (valuable subset) associated with the l 'th 1 out of K constraint as s_l , thus generic K is now defined according to the size of set s_l , i.e., $|s_l| = K_l$. We index the individual ranking constraints in a valuable set s_l by h . We know that one of the ranking constraints in each s_l must hold. We use an L -vector $e = (e_1, \dots, e_L)$, where $1 \leq e_l \leq K_l$, to denote the indices of the constraints that hold from the minimal valuable subsets. That is, we assume that constraint e_l holds in s_l for all l so that the overall restricted Job Agent Subproblem is feasible.

We now define a preprocessed scheduling instance with partially established ranking according to the satisfied elements of valuable subsets with respect to a given vector e . We can interpret this as a preprocessed disjunctive graph obtained by fixing some of the disjunctive arcs according to the members of e . For each preprocessed instance (or associated disjunctive graph) that corresponds to given e , we define a set of predecessors $B_{ij}(e)$ of each operation (i, j) as the operations that have higher priority in the partially established ranking, or that are already determined to precede operation (i, j) . In this way,

$$B_{ij}(e) = \{(i', j') : ((i', j'), (i, j)) \in \sigma(e)\},$$

where $\sigma(e)$ is the overall ranking set that is defined by the constraints that are assumed to hold and governed by the elements of e . We can now calculate constrained earliest starting time ($CEST_{ij}(e)$) for each operation (i, j) with respect to set $B_{ij}(e)$ by using the following recursive formula:

$$CEST_{ij}(e) = \max \left\{ CEST_{i,j-1} + p_{i,j-1}, \max_{\beta \subseteq B_{ij}(e)} \left\{ \min_{(i',j') \in \beta} CEST_{i',j'} + \sum_{(i',j') \in \beta} p_{i',j'} \right\} \right\}. \quad (4.17)$$

Note that the first term in the outer maximization, called *job availability*, is a

4.5. MODEL OF OTHERS

restriction due to the serial precedence relation between operation j of job i and the previous operation $(j - 1)$ of the same job. The second term, called the *machine availability*, is a tight lower bound on the completion time of the operations that are to be processed before operation (i, j) due to the ranking set $\sigma(e)$. For a given ranking $\sigma(e)$ and associated predecessor set $B_{ij}(e)$, $CEST_{ij}(e)$ defines the earliest time that the machine m_{ij} can start processing operation (i, j) .

To calculate the true lower bound on the starting times we need to consider all constraint realization profiles e . Only this way can we get the constrained earliest start times for the Job Agent Subproblem with 1 out of K constraints. Hence, we need to find the minimum of the values of $CEST_{ij}(e)$ over e :

$$CEST_{ij}(I) = \min_{e: 1 \leq e_l \leq K_l} CEST_{ij}(e)$$

We now consider inserting the CEST constraints into the Job Agent Subproblem instead of the ranking-based precedence constraints and 1 out of K constraints. These constraints are constructed as follows:

$$S_{ij} \geq CEST_{ij}(I), i = 1, \dots, N, j = 1, \dots, n_i.$$

The new Job Agent Subproblem is an alternative to problem $LR(\lambda, I)$. We use the notation $LR(\lambda, C(I))$ to denote the new Job Agent Subproblem, where $C(I)$ is the set of CEST constraints defined from I . Problem $LR(\lambda, C(I))$ has two main advantages: (1) Although it may seem to be a relaxed version of problem $LR(\lambda, I)$ this is not the case. This is due to the fact that the inner maximization term in the CEST formula (4.17) considers machine capacity constraints implicitly whereas $LR(\lambda, I)$ does not take into account the capacity limitation at all. Hence, the new Job Agent Subproblem is not dominated by the other. By limiting the starting times according to CESTs, we may achieve tighter bounds and more realistic starting times although we may lose the predetermined ranking defined in I . (2) More importantly, we are able to decompose the new problem into job-level subproblems as we were in the original Job Agent Subproblem, $LR(\lambda)$. Hence, each job can solve the corresponding job-level subproblem with the additional set of CEST constraints that are defined for that job

4.6. AN ALGORITHM FOR PROACTIVE EQUILIBRIUM FOR JSP

only. This will be useful when we link our discussion to the Proactive Equilibrium concept. (3) Computationally, $LR(\lambda, C(I))$ is much easier than its counterpart, since the job-level subproblems still satisfy the nice properties of original LR. This is not the case for $LR(\lambda, I)$, which is itself an IP due to the involvement of decision variables Y .

Since CESTs are calculated from I which is a collection of 1 out of K constraints, the CEST-versions of the propositions outlined in the previous section are valid. The next two propositions are parallel to Propositions 4.6 and 4.7, respectively:

Proposition 4.9 *For a given I , $v(LR(\lambda, \emptyset)) \leq v(LR(\lambda, C(I)))$.*

Proposition 4.10 *For a given I , $v(LRD) \leq v(LRD(C(I))) \leq v(JSP)$.*

Finally, we summarize the best case scenario for the CEST-based problem, where we luckily find the optimal CESTs, which is parallel to Proposition 4.3:

Proposition 4.11 *If $CEST_{ij}(I) = CEST_{ij}(\sigma^*)$, $i = 1, \dots, N$, $j = 1, \dots, n_i$, for an optimal ranking $\sigma^* \in \Sigma^*$, then $v(LRD(C(I))) = v(LR(0, C(I))) = v(JSP)$.*

Note again, in this unlikely case, we can close the duality gap even without multiplier search since $\lambda^* = 0$ for $LRD(C(I))$.

4.6 An Algorithm for Proactive Equilibrium for JSP

The main idea behind the PEJSP is to give the job agents the capability to consider others' existence not only by using prices but also by restricting themselves using generated constraints. The primary issue is to give the agents this capability without risking the quality of the overall solution. To achieve this, the center can in fact announce the CEST values calculated as in previous sections so that the agents may update the constraints in its *models of others*. One obvious tool for this is to have the agents restrict their internal models, which are essentially starting time constraints

4.6. AN ALGORITHM FOR PROACTIVE EQUILIBRIUM FOR JSP

collected in $\hat{\omega}_i(\phi)$, by using the CEST values announced by the center. Hence, an agent i can restrict right hand sides of the starting time constraints as follows in the update of the models:

$$\tau_{ij} \leq CEST_{ij}(I), j = 1, \dots, n_i.$$

Under the above setting, we outline a mechanism to achieve Proactive Equilibrium as follows:

1. Initialize the pricing vector, the set of extra constraints, and the CEST values. Set $\lambda^1 = 0$, $I^1 = \emptyset$, and $CEST_{ij}^1 = \sum_{j'=1}^{j-1} p_{ij'}$ (Note that these CESTs are currently redundant in the job-level subproblems because of the existing (hard) operation precedence constraints in the subproblems). Set iteration number $r = 1$.
2. Each agent inserts constraints in its *models of others* using the CEST values currently announced, i.e., locally set values of τ_{ij}^r considering the limitation $\tau_{ij}^r \leq CEST_{ij}^r$. Each agent solves the restricted job-level subproblems so as to maximize $U_i(\phi, \hat{\omega}_i(\phi), X_i) = U_i(\lambda, \tau_i^r, X_i)$ subject to $X_i \in \mathcal{X}_i(\tau_i^r)$. (This is the same as $LR_i(\lambda^r, C(I^r))$ except that CESTs in $C(I^r)$ are replaced with τ_{ijs}^r). The center may at this point construct a shop-level Lagrangean schedule, and compute a strong ranking $\sigma(\lambda^r)$. The center, in turn, calculates the corresponding Lagrangean lower bound $LB(r)$. Update the best lower bound if necessary, i.e., $LB^* = \max\{LB(r), LB^*\}$.
3. Isolate the minimal valuable subsets from the strong ranking set. Update the 1 out of K constraints and associated ranking constraints, i.e., $I^{r+1} \leftarrow I^r \cup I_{\sigma(\lambda^r)}$. Compute the corresponding CEST values, $CEST_{ij}^{r+1} = CEST_{ij}(I^{r+1})$ for announcement in the following iteration.
4. To increase efficiency, the center may also update the best UB value as follows: use a feasibility restoration routine to compute a feasible schedule from the agents' local solutions and record this value as $UB(I^{r+1})$. Update best upper bound if $UB^* = \min\{UB(I^{r+1}), UB^*\}$.

4.6. AN ALGORITHM FOR PROACTIVE EQUILIBRIUM FOR JSP

5. If a stopping condition is satisfied, stop, report the best lower bound and best upper bound (if not equal), and its corresponding schedule. Otherwise, set $r \leftarrow r + 1$ and go to Step 2.

For a PEJSP, the primary stopping condition is to check if the current values of τ^r match the starting times found in the current feasible schedule. If this is the case, we know that current collective solution is globally feasible. Since this is not guaranteed to be achievable in all cases, we can terminate using other heuristic stopping conditions.

Note that in this algorithm, the number of 1 out of K constraints and the number of associated minimal valuable subsets successively gets bigger over the iterations. Hence, the $CEST_{ij}^r$'s are monotonically increasing over the iterations, i.e., $CEST_{ij}^{r+1} \geq CEST_{ij}^r$. This could be reflected to τ values by using a straight-forward update formula:

$$\tau_{ij}^r = CEST_{ij}(I^r)$$

Here the agents directly accepts the CEST values computed by the center as the parameters of the their constraints in the models of others.

This potentially improves the Lagrangean lower bound, which in turn narrows the duality gap, defined as $DG = v(JSP) - LB^*$. Since we normally do not have $v(JSP)$, we (over)estimate the duality gap by calculating the *estimated duality gap* (EDG) defined as the difference between best UB and best LB: $EDG = UB^* - LB^*$. For this estimated measure, the algorithm provides additional improvement over the standard procedure because of the potential improvement for the UB values. The improvement is due to the fact that the CEST values restrict the UB search to more promising areas. This is stated in the next proposition:

Proposition 4.12 *Consider a feasibility restoration routine modified to accommodate provably exact ranking constraints (1 out of 1 constraints) and the CEST values. Assume that remaining operations are ranked by using a standard tie-breaking rule in the feasibility restoration routine. If $I \subseteq I'$, then $UB(I) \geq UB(I')$.*

Hence, using more ranking-based constraints and tighter CESTs in Lagrangean problem narrows the (estimated) duality gap in two directions: (1) by improving

4.7. AN ILLUSTRATIVE EXAMPLE

the Lagrangean lower bound, and (2) by improving the upper bound. This in effect allows proactive equilibrium to reduce the communication overhead by cutting down the number of iterations necessary and by closing the duality gap at termination.

4.7 An Illustrative Example

In this section, we will use an example to illustrate how the PE may be implemented for a scheduling problem. This also serves to show how the important steps of the algorithm outlined in the previous section can be implemented. We base our example on real data that we collected from an electronics facility of the Ford Motor Company. For a detailed description of the system, see [48].

In this example, we focus on one of the SMD resources, CSMD10C. We have 33 jobs and the main objective is (weighted) tardiness to reflect the real concern of the manager who is responsible from the Electronic Engine Control (EEC) product line (Weights are randomly drawn between 1 and 10). A similar presentation can be done for multiple resources, or for more complex scheduling problems such as classical job shop scheduling problems. The data for the example problem is shown in Table 4.2.

We first run our base algorithm without any additional constraints in the job level subproblems. A typical strong ranking (non-trivial ranking where all job agents do not demand same time slots) forms in a couple of iterations. In iteration 4, for example, a part of the strong ranking suggests:

$$\sigma(\lambda) = \{3 \rightarrow 33, 4 \rightarrow 33, 8 \rightarrow 33, 26 \rightarrow 33, 27 \rightarrow 33, 29 \rightarrow 33, 33 \rightarrow 17, \dots\}$$

Suppose that we have a good upper bound value (UB^*) that is either obtained using a heuristic or as an output of a feasibility restoration routine updated over iterations. In this example, we will assume that we know the best upper bound value that is obtained running the base case for 50 iterations, $UB^* = 1084$. Note that we can start with a rather loose upper bound which will be less efficient to create the valuable subsets early in iterations. Updating UB^* for better values will allow us to

4.7. AN ILLUSTRATIVE EXAMPLE

Table 4.2: Data for the example: Single machine, 33 jobs

Job (i)	W_i	d_i	p_i
1	1	43	4
2	10	43	2
3	7	1	5
4	8	1	3
5	5	68	1
6	8	67	11
7	2	44	2
8	2	1	2
9	7	67	4
10	5	67	2
11	5	45	1
12	9	44	1
13	5	67	2
14	2	68	1
15	1	1	3
16	7	68	1
17	3	22	25
18	2	44	1
19	6	42	3
20	4	44	1
21	10	43	2
22	6	67	2
23	6	90	2
24	5	69	1
25	6	92	1
26	3	1	2
27	9	1	2
28	8	42	2
29	2	1	1
30	9	67	2
31	3	66	3
32	9	44	1
33	8	6	20

4.7. AN ILLUSTRATIVE EXAMPLE

construct more and smaller valuable subsets and also to prove some ranking among the jobs which we cannot prove with a looser UB tardiness of 1084. When we form the problem $LRO(\lambda, \sigma^c(\lambda))$ with an additional constraint stating that the objective should be less than or equal to 1084, then the problem turns out to be infeasible. To find multiple minimal valuable subsets of strong ranking, ideally we would run an efficient IIS algorithm (Note that problem is still infeasible with any IIS that we find). We will however imitate the working of such an algorithm based on addition and deletion of reverse ranking constraints from $LRO(\lambda, \sigma^c(\lambda))$ for illustrative purposes.

Suppose that we start with the core constraints and insert the precedence constraint corresponding to $17 \rightarrow 33$. The problem LRO with this constraint is infeasible i.e., $v(LR(\lambda, 17 \rightarrow 33)) = 1157 > UB^* = 1084$. This is an example of a 1 out of 1 constraint: We can safely state that in an optimal solution job 33 precedes job 17. Then a ranking based precedence constraint that can be put is $33 \rightarrow 17$. This also allows us to update CEST of job 17: $CEST_{17} = 21$.

We next consider the following reverse-ranking constraints: $s' = \{33 \rightarrow 3, 33 \rightarrow 4, 33 \rightarrow 8\}$. This problem is again infeasible, $(v(LR(\lambda, s'))) = 1169 > UB^*$, but is this set minimal? The answer is “No” since the problem LRO is still infeasible even with $s'' = \{33 \rightarrow 3, 33 \rightarrow 4\}$. Hence $(s')^c$ is not a minimal valuable set, but what about $(s'')^c$? The answer is positive, since we cannot show that the problem is infeasible when we insert either $33 \rightarrow 3$ or $33 \rightarrow 4$. This means that, we are sure that either job 3 or job 4 precedes job 33 (or both), but we *currently* cannot prove which one(s). Nevertheless, this type of ranking constraints are useful since they allow us to update CESTs safely even if we do not have information about the exact ranking. To see this, note that s'' consists of the constraints fixing the directions from a single job (33), or alternatively $(s'')^c$ puts a job before 33 either way. Hence, job 33 should at least wait for $\min\{p_3, p_4\}$ time periods, which updates $CEST_{33} = 4$. If we continue with the normal price-update procedure (improving the lower bound for better price vector), we will see that job 3 is sure to precede job 33 since, with the new λ , $v(LR(\lambda, 3 \rightarrow 33)) = 1104 > UB^*$. This further improves the CEST of 33 by setting it to 6. Further iterations shows that job 4 must precede job 33 too, and $CEST_{33} = 9$. Note that all these updates also affect the jobs coming after 33: Job

4.7. AN ILLUSTRATIVE EXAMPLE

17, for example, now has a CEST of 29.

Another minimal valuable set is $\{26 \rightarrow 33, 27 \rightarrow 33\}$. This allows us to update $CEST_{33} = 9 + \min\{p_{26}, p_{27}\} = 9 + 2 = 11$. By observing that jobs 26 and 27 have same processing time and job 27 has smaller due date, one can heuristically put 27 before 33 permanently, by fixing $27 \rightarrow 33$. To tighten job 17's CEST more, we show that either $8 \rightarrow 17$ or $15 \rightarrow 17$ in an optimal solution, and set $CEST_{17} = 29 + \min\{p_8, p_{15}\} = 31$. After this, we were able to show that job 27 precedes job 17 which leads us $CEST_{17} = 33$. Further IIS analysis shows that either job 3 or job 33 precedes job 6 ($CEST_6 = 6$), either job 27 or job 33 precedes job 6 ($CEST_6 = 8$), and job 6 precedes job 17, $CEST_{17} = 44$.

More IISs and respective minimal valuable subsets of strong rankings found through this process showed that we can fix $7 \rightarrow 6, 9 \rightarrow 6, 10 \rightarrow 6, 11 \rightarrow 6, 23 \rightarrow 17, 25 \rightarrow 17, 33 \rightarrow 15, 33 \rightarrow 20, 33 \rightarrow 21$, and $33 \rightarrow 28$. As expected, further updates on CEST values are possible: $CEST_6 = 17, CEST_{15} = 21, CEST_{17} = 44, CEST_{20} = 21, CEST_{21} = 21, CEST_{28} = 21$.

Now we have different alternatives: (1) We can either insert 1 out of K constraints (note in some cases $K = 1$) to the LR problem although this ties the job-level subproblems. (2) We can insert CEST constraints to the respective job-level subproblems and continue with the price updates. An advantage of this is as we mentioned before that it keeps the distributed structure as it is (i.e., independent job subproblems). (3) As a brute-force method, we can insert both of these constraint types to the LR problem at the same time. This can be thought as a benchmark case since it represents the tightest formulation possible through our analysis. For illustration purposes, we consider inserting all CEST constraints and one out of K ranking constraints that is almost sure not to cut off the optimal solution. That is, in our test case we have both CEST and one-way ranking constraints added to the LR problem.

Recall that for any given λ , the new problem has both better lower and upper bound values as compared to our base case. With starting 0 prices, for example, the new model gives a lower bound of 420 and upper bound of 4723 (Compare it with the base case, where LB is 181, and UB is 6039).

To see if the constructed CEST and ranking constraints improve the convergence

4.7. AN ILLUSTRATIVE EXAMPLE

and the quality of the base model, we ran the base algorithm for 10 iterations, and then we insert the constraints, and continue the price updates with the new model. Recall that for any given λ , the new problem has both better lower and upper bound values as compared to our base case. In iteration 11, where both are using the same price vector, we have a lower bound significantly tighter than that of the base case (584 versus 563, 3.6% improvement). We also drastically improve the upper bound too: (1203 versus 1576) in that iteration. When we continue price updates with the tighter model, after certain number of iterations, the improvement for LB diminishes, and the LBs for the tighter model unexpectedly become smaller than that of the looser model (base case). This is depicted in Figure 4.1. However, the improvement obtained for the upper bound is more consistent, as we can see in Figure 4.2, better upper bounds (hence better feasible solutions) are found with the tighter LR model. We even improve our best upper bound (UB^*) with the new model ($UB^* = 1075$), whereas the base model yields the same value (1084). If we measure the difference on the relative duality gap $((UB - LB)/UB)$ for each iteration, we see that the new model narrows the gap at least 7 percentage points with a maximum improvement of 19 points except the last 13 iterations where the difference is very small but negative. Another measure is the total number of conflicts among the jobs for the time slots (i.e., the total number of jobs that request a certain time slot less the capacity of 1) that the feasibility restoration routine has to clear is much less with the tighter formulation (compare 37 versus 42). This shows that using model of others in the context of proactive equilibrium has the potential to lessen the conflicts among the jobs. This result, although limited, supports our vision that considering some sort of model of others in each agent subproblem will diminish the conflicts among the agents.

These results show that for a given price vector, the CEST and ranking based model certainly improves the base case. However, this does not always translate into a consistent improvement (for better lower bound and/or upper bound per iteration basis) in the long run, i.e., large number of iterations. However, for a limited small number of iterations, the CEST/ranking-based formulation has a great potential to improve the base model.

4.7. AN ILLUSTRATIVE EXAMPLE

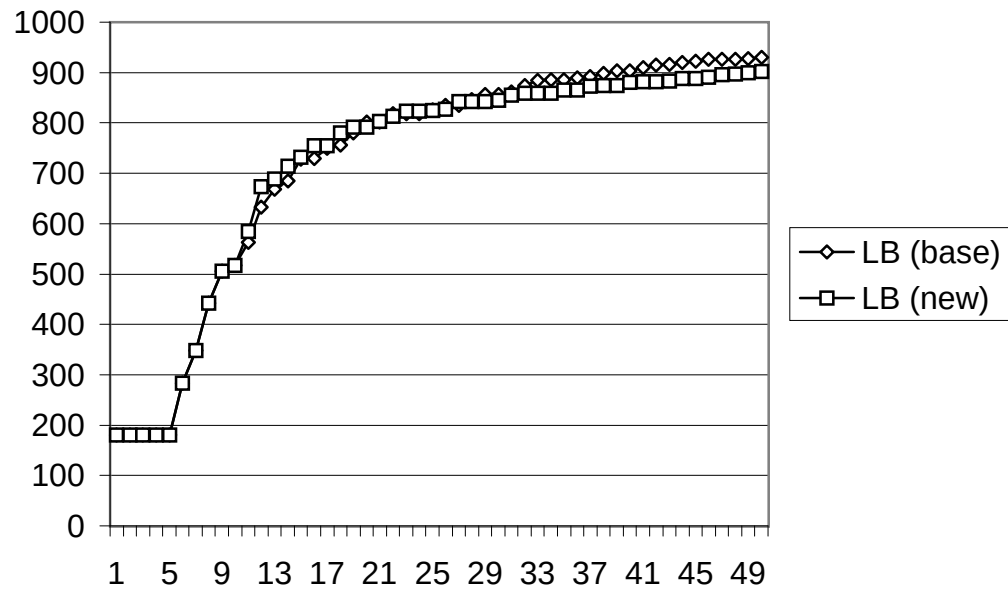


Figure 4.1: Lower bounds computed by using the base model and the model with the CEST/ranking-based precedence constraints

4.7. AN ILLUSTRATIVE EXAMPLE

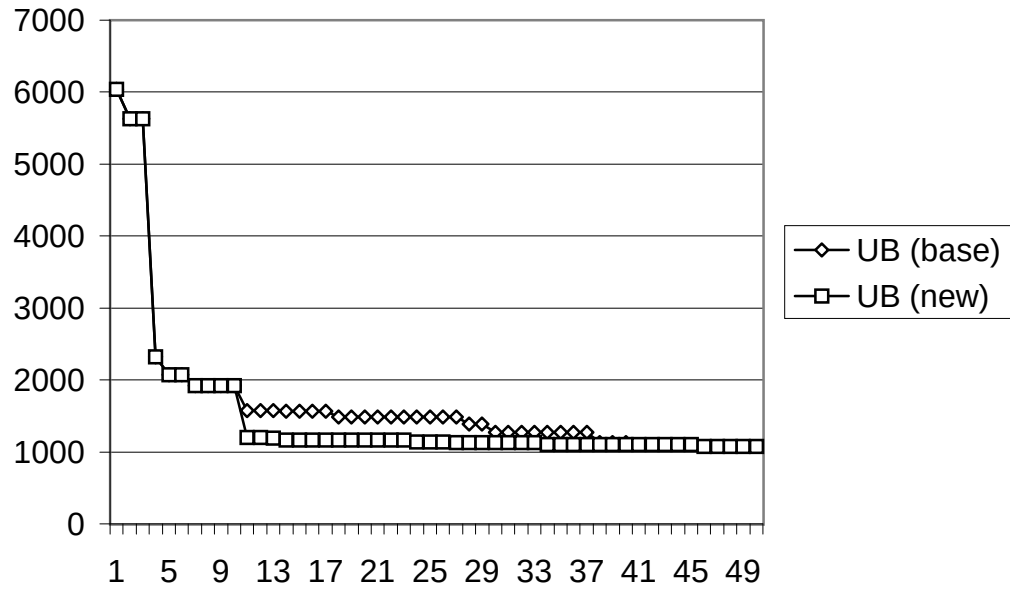


Figure 4.2: Upper bounds computed by using the base model and the model with the CEST/ranking-based precedence constraints

4.8 Conclusions

In this study, we analyzed the theoretical basis for an improved equilibrium concept for distributed decision making, with a special focus on the distributed job shop scheduling. We outlined the main requirements for such an implementation. We showed that there are opportunities for improvement in both final solution quality, for amount of communication, and for closeness to the equilibrium. However, we focused on using one of the sources of information that we mentioned in the introduction, which is information gathered over the iterations of a specific execution of a distributed algorithm. We left the use of *a priori* information for a future study. We believe that using both information (*a priori* and over the iterations) should help improve the properties outlined in this study.

Chapter 5

Incentive Compatible Scheduling

5.1 Introduction

In any multi-agent (or distributed) environment, the involvement of the agents in the decision making process is crucial. If agents are not provided with proper incentives, they may not perform in a way that is aligned with the overall system goals. In this study, we analyze this important dimension of distributed decision making known as *incentive compatibility*. The term refers to the issues of providing a reward mechanism for decision agents so as to align their individual (local) interests with system-wide (global) objectives. In spite of the apparent importance of this topic, it has been overlooked in most of the decentralized approaches to optimization. It has been generally assumed that agents always have the good will to cooperate and coordinate their decisions, and they provide all private information truthfully. Incentive compatibility deals with these issues explicitly in the design stage of the decision mechanisms. In this study, we concentrate on ways of aligning local interests with global objectives and have them communicate their private information truthfully.

To address the incentive compatibility issues, we consider a *schedule selection game*. In this environment, there are candidate feasible schedules posted for consideration, and there are job agents that each represents the interest of a particular job. The agents possess private information about the true benefits of the schedules to their individual interests. The agents are to provide a valuation of each schedule

5.1. INTRODUCTION

and this information will be used to identify a final schedule. Since each schedule allocates resources differently, job agents could have drastically different preferences over the schedules under consideration. More importantly, a fair schedule selection (resource allocation) relies on each agent’s truthful reporting of their valuation. Thus, a primary issue is to provide the agents with proper incentives that would elicit their true preferences so that the overall best schedule is selected. To this end, we devise a mechanism that produces optimal solutions regardless of the fact that individual preferences are known only by each individual, while various forms of manipulations or misrepresentations are possible.

Designing an incentive compatible procedure in the environment such as the above can be viewed as a *mechanism design problem*, a subject of extensive study in the microeconomics and game theory literature [59]. We will focus on a specific subarea of this literature known as the *direct revelation mechanisms*. Similar problems are studied in many different contexts and applications. The origin of all these studies is the “optimal control problem” of a central authority who tries to control the agents in an organization. The problem was first posed and studied as an incentive problem by Groves in the early seventies [33, 34]. A reflection of this mechanism on a resource allocation problem in a decentralized organization can be found in [37]. These procedures have been referred to as *Groves mechanisms* or *Groves incentive schemes*. In a different context, Groves studies the problem of coordinating economic agents’ decisions considering external effects [35]. This line of research goes back to Vickrey’s seminal work on “counterspeculation” and design of second-price auction mechanisms [91]. Although there are many applications of Groves mechanisms in the economics literature (see, e.g., Brewer and Plott [11]), applications to modeling and optimization did not exist until more recently. A recent implementation of this sort considers an incentive-compatible budget allocation problem for site decontamination projects [21]. A general treatment of Groves mechanisms with a special focus on Artificial Intelligence oriented Multi-agent systems can be found in [79].

Groves [36] proposes a related but different approach around the same time he developed Groves mechanisms. In this study, the presence of central control is no longer needed. He analyzes a collective “public choice” problem where agents try

5.2. THE SCHEDULE SELECTION GAME

to implement a “socially efficient” (or optimal) public project. In this chapter, we follow an approach closely related to this latter line of research where the mechanism involves collective decision making and self-coordination of job agents rather than a center-based control approach (we provide a brief discussion on this issue in Section 5.4). As common in all previous incentive compatibility studies, we will assume the existence of a transferable commodity which is usually represented by “money.” We will consider self-interested agents who try to maximize their local (net) utility as a combined function of their original utility and the amount of reward (money) that the mechanism allocates. We interpret the original utility as “profits” that an agent makes when a certain schedule is implemented. Without loss of generality, this assumption simplifies the discussion of the mechanism since the original utility and the transfer have the same unit (“money” or dollars). In Section 5.4, we discuss the implications and possible alternatives of this assumption. In Section 5.2, we first outline a direct revelation mechanism for the schedule selection game, and then discuss various properties and relevant game theoretic results such as dominant strategy equilibrium. We then investigate related issues such as the generation of candidate schedules and the quality of the selected schedule through the mechanism. We give an example to illustrate a typical implementation of the approach in a real-life scheduling environment in Section 5.3. The last section provides conclusions and future research topics.

5.2 The Schedule Selection Game

We first define some notations used throughout the chapter: For a generic vector n -vector x , x_i denotes the i th element of the vector, i.e., $x = (x_1, \dots, x_n)$, and x_{-i} denotes $n-1$ -vector of the elements of x without x_i , i.e., $x_{-i} = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$. We use this notion of “others” ($-i$) to represent the other agents’ actions/decisions with respect to the agent under consideration, agent i . Similarly, for a given set $X \equiv \cup X_i$, we will use X_{-i} to denote the set X without X_i . Thus, we can interpret $-i$ in the subscript as the original vector or set without the i th component, i.e., $-i \equiv \mathcal{N} \setminus \{i\}$.

5.2. THE SCHEDULE SELECTION GAME

The *schedule selection game* can be described as follows. There is a set of job agents $\mathcal{N}$, indexed by $i = 1, \dots, N$, and a set of feasible shop schedules Y known to all agents. Each schedule is a complete and feasible job-shop schedule where each job receives a particular block of resource time for each of its operations. A job agent represents one and exactly one job in the schedule. The impact (valuation) of any particular schedule to a job agent is private information only known to the agent. The job agents must collectively choose an *outcome* consisting of (1) a *shop schedule* y from the set of feasible schedules Y , and (2) *transfers* of a numeraire commodity (“money”) from/to all agents, $\{\delta_i\}_{i \in \mathcal{N}}$. Here δ_i denotes the amount of money transferred to (if positive) or from (if negative) agent i . We assume that the preferences of each job agent over the outcomes are summarized by a utility function u_i . We also assume that an agent i only cares about the transfer δ_i and the profit π_i he receives from the schedule y :

$$u_i(y, \delta_i) = \pi_i(y) + \delta_i. \quad (5.1)$$

Note that the utility u_i of agent i does not depend on other agents’ transfers, δ_{-i} . Thus, agent i prefers outcome (y, δ) with schedule y and transfer vector $\delta \equiv (\delta_1, \dots, \delta_N)$ to another outcome (y', δ') if and only if $u_i(y, \delta_i) \geq u_i(y', \delta'_i)$. For a classical JSP with total weighted tardiness objective, the profit π_i of an agent i can be interpreted as negative weighted tardiness costs that is converted to monetary units. In the following discussion, we will use the generic term “profit” while keeping in mind that it can represent a variety of objectives including weighted tardiness. Simply stated, the schedule selection game is a decision procedure (a mechanism). Through the use of proper transfer functions (incentives) agents will always reveal their true preference while the collection of all agents’ decisions correspond to the best solution possible for all.

5.2.1 Problem Analysis

We now discuss several important properties of the *schedule selection game* using a game theoretic framework. We assume that a job agent’s preferences are known only by the agent himself. In a manufacturing context, the preferences may be motivated

5.2. THE SCHEDULE SELECTION GAME

by delivery requirements or contractual agreement specific to the job, the performance evaluation a product manager is subject to, or any other constraints that may be considered “local.” Any decision procedure that is responsive to individual preferences will require some type of communication or signaling of preference information.

We first define the schedule selection game as an N -person noncooperative game with incomplete information. A *mechanism* Γ is a collection $(\{M_i\}_{i \in N}, h(\cdot))$ where M_i is a set of alternative messages μ_i for agent i , and $h(\cdot)$ is an *outcome function*. Function $h(\cdot)$ specifies for every joint message $\mu \equiv (\mu_1, \dots, \mu_N)$, (1) a collective schedule y , and (2) a transfer profile δ , that is $h(\mu) = (y, \delta)$. In the following discussion, we separate the outcome function into two parts, i.e., $h(\mu) = (y(\mu), \delta(\mu))$, where $y(\cdot)$ and $\delta(\cdot) \equiv (\delta_1(\cdot), \dots, \delta_N(\cdot))$ denote the *schedule selection function* and *transfer function*, respectively. Thus, we have a game composed of N players corresponding to the job agents. The strategy set of player i is the message space M_i of agent i , and the preferences of player i over joint strategies μ (messages) are defined by the utility function u_i and by the outcome function $h(\cdot)$, i.e., player i prefers μ over μ' if and only if $u_i(h(\mu)) \geq u_i(h(\mu'))$. This defines an N -person non-cooperative game of incomplete information since each player’s preferences are known only to himself.

Typically, in a game theoretic framework, the goal of a mechanism is to choose a particular outcome using the outcome function $h(\cdot)$ for a particular *realization* of agents’ utility functions, say $\{\pi_i\}_{i \in N}$. From a system optimization point of view, a desirable result would be to choose an *optimal*, or *socially efficient*, schedule y^* , i.e., find a schedule y^* in Y such that $\sum_i \pi_i(y)$ is maximized. We say that mechanism Γ with outcome function $h(\cdot) = (y(\cdot), \delta(\cdot))$ *implements* an optimal schedule y^* if there is an *equilibrium* joint message μ^* of the game induced by Γ whose outcome is y^* . Note that the implementation of a particular outcome depends on the equilibrium concept used. Although our main focus is on finding the collectively best schedule rather than the actual transfer amounts, we should note a more demanding implementation is to select an *ex post efficient* outcome which not only requires an optimal schedule but also a zero total transfer of money. A well-known result for the utility functions introduced in (5.1) is summarized in the following:

Proposition 5.1 (*Mas-Colell et al. [59]*) *For given quasilinear utility functions as*

5.2. THE SCHEDULE SELECTION GAME

in (5.1), (y^*, δ^*) is *ex post efficient* (Pareto optimal) if and only if $\sum_i \delta_i^* = 0$ and $y^* = \arg \max_{y \in Y} \sum_i \pi_i(y)$.

Since the specific form of the outcome function is a mechanism design choice, we seek outcome functions that implement a particular schedule, more specifically an optimal one. From this point of view, we are particularly interested in *direct revelation mechanisms* which normally involve two stages: (1) In the first stage, each agent is asked to reveal his true profit from the schedules, i.e., each $\mu_i(y)$ is the value reported by i that supposedly represents the profit obtained by agent i if y is selected. (2) After all agents report their values, the outcome function $h = (y(\cdot), \delta(\cdot))$ chooses the maximizer of the the total reported profit, i.e.,

$$y^*(\mu) = \arg \max_{y \in Y} \sum_i \mu_i(y), \quad (5.2)$$

assuming that all the agents report their true profits, i.e., $\mu_i(y) = \pi_i(y)$ for all y , and the payments are made according to transfer function $\delta(\mu)$.

Note that if all agents truthfully report their profits, then the schedule selection function (5.2) finds an optimal schedule y^* in Y . In fact, this requirement of having the agents communicate true profits is the essence of direct revelation mechanisms. However, because of the dispersed information and individual preferences over schedules, it may not be in any agent's best-interest to "tell the truth". For example, by reporting falsely his valuations of the schedules y , an agent may be able to secure a larger transfer than necessary to compensate him for any change in y that his false reporting causes.

Thus, without a proper *incentive scheme* that induces "truth telling" behavior, an agent may lie about his profits, with the intent to maximize his local utility, $u_i(y^*(\mu))$. Hence, the primary focus in designing direct revelation mechanisms is on constructing a proper incentive scheme. Such schemes are characterized by carefully designed *transfer functions*, $\delta(\cdot)$, which is a part of the outcome function. With such an incentive scheme, the agents voluntarily report their true profits. To formally study the aforementioned issue of *incentive compatibility*, i.e., inducing truth telling as an equilibrium behavior, we need to define a specific equilibrium for the game induced

5.2. THE SCHEDULE SELECTION GAME

by the direct revelation mechanism. In the following, we summarize the concept of dominant strategy equilibrium for the mechanism and analyze its properties.

5.2.2 Dominant Strategy Equilibrium

Dominant strategy equilibrium states that each player plays his best strategy that is individually best for him regardless of the strategy chosen by any other players.

Definition 5.1 (Dominant Strategy Equilibrium) *For mechanism Γ with message space $\{M_i\}_{i \in N}$ and outcome function $h(\cdot)$, a joint message $\mu^* = (\mu_1^*, \dots, \mu_N^*)$ constitute a dominant strategy equilibrium if, for every i ,*

$$u_i(h(\mu_i^*, \mu_{-i})) \geq u_i(h(\mu_i, \mu_{-i})) \quad \forall \mu_i \in M_i, \quad \forall \mu_{-i} \in \mathcal{M}_{-i}$$

where $\mu_{-i} \equiv (\mu_1, \dots, \mu_{i-1}, \mu_{i+1}, \dots, \mu_N)$ and $\mathcal{M}_{-i} \equiv \prod_{i' \neq i} M_{i'}$.

Thus, if a dominant strategy equilibrium exists for a mechanism, then each player would have a strong incentive to play his dominant strategy. For a direct revelation mechanism with proper incentives, this means that every job agent has an incentive to report his true valuation of the schedules no matter what the other agents report since doing so maximizes his own (true) utility, i.e., $\mu_i^*(y) = \pi_i(y) \quad \forall y \in Y$.

Assuming the schedule selection (5.2) of an outcome function, agent i , for every y , solves the utility maximization problem

$$\mu_i^*(y) = \arg \max_{\mu_i \in M_i} u_i(y^*(\mu_i, \mu_{-i}), \delta_i(\mu_i, \mu_{-i}))$$

or

$$\mu_i^*(y) = \arg \max_{\mu_i \in M_i} \pi_i(y^*(\mu_i, \mu_{-i})) + \delta_i(\mu_i, \mu_{-i})$$

for any $\mu_{-i} \in \mathcal{M}_{-i}$. Note that through messages the agent is able to affect both the final schedule selection and his transfer which in turn influence the total payoff to agent i . As mentioned above, it is the goal of the transfer function $\delta(\cdot)$ to provide incentives to the agents to reveal their true valuations regardless of what the other agents report.

5.2. THE SCHEDULE SELECTION GAME

Consider the case where transfers are all zero, i.e., $\delta(\mu) = 0 \forall \mu \in \mathcal{M}$. In this case, $u_i(y, \delta_i = 0) = \pi_i(y)$. Thus, without transfers, each job agent will try to maximize his original profit when he is asked to report his valuation of the schedules. As a self-interested agent, job agent i will not tell the truth by using the following reporting rule (assuming minimum profit of 0 from any schedule):

$$\mu_i^*(y, \delta_i = 0) = \begin{cases} L_i & \text{if } y = \arg \max_{y' \in Y} \pi_i(y') \\ 0 & \text{otherwise.} \end{cases} \quad (5.3)$$

where L_i is a (large) number which may or may not be equal to true value $\pi_i(y)$. Hence, without transfer, the agents tend to inflate the profit for their local best schedule, and report minimum possible profit (0) for all others. Using these reported values, it is clear that the outcome function as defined in (5.2) and $\delta(\cdot) = 0$ cannot implement an optimal schedule. In fact, the following proposition is a special case of a result known as Gibbard-Satterthwaite Impossibility theorem [36, 59].

Proposition 5.2 *A direct revelation mechanism Γ with schedule selection function defined by (5.2) cannot implement an optimal schedule in dominant strategies if transfers are not allowed.*

In fact, a specific form of transfer function known as *Groves incentive scheme* (Groves [34, 36]) is sufficient to guarantee the implementation of an optimal schedule in dominant strategies as outlined in the following proposition (Note how the net utility functions of the agents and the joint profits are aligned through schedule selection and transfer functions).

Proposition 5.3 *A direct revelation mechanism $\Gamma = (\{M_i\}_{i \in \mathcal{N}}, h = (y^*(\cdot), \delta^*(\cdot)))$ with schedule selection function defined by (5.2) implements an optimal schedule in dominant strategies if, for all $i \in \mathcal{N}$,*

$$\delta_i^*(\mu) = \sum_{i' \neq i} \mu_{i'}(y^*(\mu)) - R_i(\mu_{-i}) \quad (5.4)$$

where $R_i(\cdot)$ is an arbitrary function of μ_{-i} .

5.2. THE SCHEDULE SELECTION GAME

Proof: The proof is a modified version of the argument used in [36]. To prove the proposition, it is sufficient to show that the agents reveal their true profits in a dominant strategy equilibrium, i.e., for every $i \in \mathcal{N}$, $\mu_i^*(y) = \pi_i(y) \forall y \in Y$, where μ^* are dominant strategy equilibrium messages. We first denote the schedule that would be selected if $\mu_i(y) = \pi_i(y) \forall y \in Y$ as $y^*(\mu_i = \pi_i, \mu_{-i})$ when the others' joint message is shown by μ_{-i} . From definition of schedule selection function, we recall that $y^*(\mu_i = \pi_i, \mu_{-i})$ maximizes $\pi_i(y) + \sum_{i' \neq i} \mu_{i'}(y)$ over all schedules $y \in Y$. Then,

$$\pi_i(y^*(\mu_i = \pi_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i = \pi_i, \mu_{-i})) \geq \pi_i(y) + \sum_{i' \neq i} \mu_{i'}(y) \forall y \in Y. \quad (5.5)$$

The inequality (5.5) holds for any y in Y including $y^*(\mu_i, \mu_{-i}) \in Y$ for any $\mu_i \in M_i$. Thus,

$$\begin{aligned} \pi_i(y^*(\mu_i = \pi_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i = \pi_i, \mu_{-i})) &\geq \\ \pi_i(y^*(\mu_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i, \mu_{-i})) &\forall \mu_i \in M_i. \end{aligned} \quad (5.6)$$

Note that the left hand side of inequality (5.6) is the utility of i from $y^*(\mu_i = \pi_i, \mu_{-i})$ plus the term $R_i(\mu_{-i})$:

$$\pi_i(y^*(\mu_i = \pi_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i = \pi_i, \mu_{-i})) = u_i(y^*(\mu_i = \pi_i, \mu_{-i}), \delta^*) + R_i(\mu_{-i})$$

Similarly, for the right hand side of the inequality,

$$\pi_i(y^*(\mu_i, \mu_{-i})) + \sum_{i' \neq i} \mu_{i'}(y^*(\mu_i, \mu_{-i})) = u_i(y^*(\mu_i, \mu_{-i}), \delta^*) + R_i(\mu_{-i})$$

Then,

$$u_i(y^*(\mu_i = \pi_i, \mu_{-i}), \delta^*) + R_i(\mu_{-i}) \geq u_i(y^*(\mu_i, \mu_{-i}), \delta^*) + R_i(\mu_{-i})$$

or, since R_i is independent of μ_i ,

$$u_i(y^*(\mu_i = \pi_i, \mu_{-i}), \delta^*) \geq u_i(y^*(\mu_i, \mu_{-i}), \delta^*) \forall \mu_i \in M_i.$$

5.2. THE SCHEDULE SELECTION GAME

This holds for every i and for every μ_{-i} which proves the result. $\diamond$

Note that any joint message μ with $\mu_i(y) = \pi(y) + L_i \forall i$ where L_i is now a constant does not change the selection of an optimal schedule. Hence, the above proposition holds even if the agents increase or decrease their respective profits by a constant amount. A direct revelation mechanism Γ with outcome function $h(\cdot) = (y^*(\cdot), \delta^*(\cdot))$ (see equations (5.2) and (5.4) respectively) is known as *Groves mechanism* (Groves [33, 36]). In fact, a specific version of function $R_i(\cdot)$ was first proposed by Clarke [18] and then extensively used in direct revelation mechanisms developed for different applications (see, e.g., [21]):

$$R_i(\mu_{-i}) = \sum_{i' \neq i} \mu_{i'}(y_{-i}^*(\mu_{-i}))$$

where $y_{-i}^*(\cdot)$ denotes the outcome that would be used without agent i 's messages, i.e.,

$$y_{-i}^*(\mu_{-i}) = \arg \max_{y \in Y} \sum_{i' \neq i} \mu_{i'}(y)$$

Here, $R_i(\cdot)$ can be interpreted as a charge equal to the total profit that would be possible if agent i did not participate in the decision making process at all. With this setting of function R_i , the total transfer for i can be written as follows:

$$\delta_i^*(\mu) = \sum_{i' \neq i} \mu_{i'}(y^*(\mu)) - \sum_{i' \neq i} \mu_{i'}(y_{-i}^*(\mu_{-i})).$$

We will call Groves mechanisms with Clarke transfer functions as Groves-Clarke mechanisms. Note that agent i 's transfer δ_i in a Groves-Clarke mechanism is zero if his announcement does not change the schedule selection relative to what would be optimal for agents $\mathcal{N} \setminus \{i\}$, i.e., if $y_{-i}^*(\mu_{-i}) = y^*(\mu)$. It is negative if his participation causes the others to choose a different schedule, where the agent is called *pivotal*. In this case, he pays the difference he makes on the others' profits. Thus, in Groves-Clarke mechanism, all agents end up paying zero or some positive amount of money (another AI-oriented application of Groves-Clarke mechanism can be found in Sandholm [79]).

5.2. THE SCHEDULE SELECTION GAME

We showed that if the agents use a direct revelation mechanism with the outcome function defined by (5.2) and (5.4), they will be able to coordinate their individual preferences and select an overall optimal schedule. However, we should note that there are many issues to cover for a successful implementation of such a direct revelation mechanism. We will analyze some of these in the following sections.

5.2.3 Balancing the Budget

An issue in a transfer-based mechanisms is to have the total transfers satisfy certain properties. For example, one mechanism designer may need to have a nonnegative total transfer or nonpositive total transfer, or even more demanding, zero total transfer (no waste of money), i.e., fully balanced budget (recall that the latter is necessary for *ex post* efficiency). For example, in a centerless Groves mechanism such as ours, one requirement might be that no outside source of financing is allowed, i.e.,

$$\sum_i \delta_i \leq 0, \quad (5.7)$$

This is easily satisfied by the Clarke transfer function outlined in the previous section. In fact, this condition is satisfied as long as

$$\sum_i R_i(\mu_{-i}) \geq (N-1) \sum_{i'} \mu_{i'}(y^*(\mu)) \quad \forall \mu \in \mathcal{M}.$$

A more restrictive and sufficient (but *not* necessary) condition for a Groves mechanism to satisfy condition (5.7) is that

$$R_i(\mu_{-i}) \geq \sum_{i' \neq i} \mu_{i'}(y_{-i}^*(\mu_{-i})) \quad \forall \mu_{-i}.$$

From this, we see that the Clarke transfers set R_i 's to their minimum values thereby guaranteeing nonpositive total transfer.

In this case, all agents are charged or taxed by some nonnegative amount, usually total transfer being strictly negative. However, a more demanding requirement is that of having the total transfer of zero. This requirement is called *balanced budget*

5.2. THE SCHEDULE SELECTION GAME

and formally denoted by the constraint

$$\sum_i \delta_i = 0.$$

Note that, in this case, some δ_i will have to be positive, i.e., some agents will be paid a positive amount. Since the total transfer of money is zero, we can interpret this as having some of the agents pay to the other agents; no money enters or leaves the system (ideally, the *gainers* of the collective schedule selection should compensate the aggregate loss of *losers* of the coordination). However, it is shown that, in general, there is no direct revelation mechanism that implements an optimal solution in dominant strategies and satisfies balanced budget constraint for every possible message profile [32]. In the following, we present a condition which is sufficient to have a balanced budget with a Groves mechanism.

Proposition 5.4 (Groves [36]) *If the transfer function $\delta^*(\cdot)$ is set as*

$$\delta_i^*(\mu) = \sum_{i' \neq i} \mu_{i'}(y^*(\mu)) - R_i(\mu_{-i})$$

where $R_i(\mu_{-i})$ are functions such that

$$\sum_i R_i(\mu_{-i}) = (N-1) \sum_{i'} \mu_{i'}(y^*(\mu)) \quad \forall \mu \in \mathcal{M}, \quad (5.8)$$

then the budget is balanced, i.e., $\sum_i \delta_i^*(\mu_i) = 0$.

The difficulty in balancing the budget comes from the fact that in general there do not exist functions $R_i(\cdot)$ satisfying the condition (5.8) for every μ . Thus, the presence of private information means that the agents must either accept some waste of money ($\sum_i \delta_i \leq 0$) as in the Clarke transfers, or give up on finding an optimal schedule all the time.

However, we can overcome this if we introduce an extra agent whose preferences are known, or who has no preferences over the schedules. Let “agent 0” denote this extra agent. This agent’s utility might be set as $u_0 = t_0(\mu)$ which summarizes that

5.2. THE SCHEDULE SELECTION GAME

he only cares the transfer but not the schedule selection (agent 0 can be interpreted as a central authority who tries to maximize his total transfer like a seller). In this case, if we set $\delta_0^*(\mu) = -\sum_{i \in \mathcal{N}} \delta_i(\mu)$, then the Groves mechanism both satisfies the balanced budget requirement and selects the optimal schedule. Agent 0 collects all the payments from the agents which guarantees a nonnegative profit for him. In fact, we can interpret this agent as “the center” who can design and impose a mechanism, and manage the schedule selection and transfers among the agents.

5.2.4 Participation and Asynchronous Decision Making

In the previous sections, we showed that each agent has an incentive to truthfully report his local profit when asked. However, we did not question if the agents have an incentive to *participate* in the “schedule selection” process in the first place. We implicitly assumed that all agents are required to participate in the game without other alternatives. If the agents are free to participate while full participation is necessary for overall system performance, then the *individual rationality* or *participation* constraints should be considered in the design stage so that the agents are provided with the incentives to participate in the process.

We first analyze if total participation is necessary for the ultimate selection of the optimal schedule (in Y). This is also an important issue if the scheduling game is to be implemented asynchronously, say on an Internet environment. We first note that even if an agent chooses not to participate, the schedule selected without his participation will include his job. By design, a schedule selected with full or partial participation will be a complete and feasible schedule inclusive of all operations of all jobs. Since the agents’ strategies are defined as messages, not as individual scheduling decisions, the only part missing from a nonparticipant is his messages. In this case, the agent does not pay or receive any transfer but is bound to accept the schedule that other agents collectively choose without his input. Thus, we treat the non-participation of an agent as the case where he reports zero profits for all schedules in Y , i.e., he is indifferent for all schedules.

This fact allows us to analyze a form of asynchronous decision making in the

5.2. THE SCHEDULE SELECTION GAME

schedule selection game. For instance, the schedule selection function can be activated any time to find the current best schedule with potentially partial messages. Without waiting for all of the messages from all of the agents, the outcome function can select a schedule with respect to the currently reported values from currently participating agents. More importantly, once having submitted his message, there is no need for the agent to revise his message, regardless of which other agents will end up participating. This results in two properties potentially useful for asynchronous applications: (1) the agents do not have to report their profit functions simultaneously, and (2) the agents do not have to report their profits for all the schedules, i.e., they may report on a subset of the schedules.

Before starting a formal analysis, we recall that with a Groves-Clarke mechanism if *all* agents report their profits for *all* schedules, an optimal schedule in set Y will be selected. We consider this the *full participation case*. Consider an alternative, *partial participation case*, where a subset of agents, say $I \subset \mathcal{N}$, choose not to participate while the rest, denoted by $I' = \mathcal{N} \setminus I$, report for *all* of the schedules. In the next proposition, we show that if none of the agents in I is charged a positive amount of money in the full participation case, then their non-participation does not affect the schedule selected in the partial participation case.

Proposition 5.5 *If the total transfer of the agents in set I is zero under the full participation Groves-Clarke mechanism, then the scheduling selection function in the partial participation case selects an optimal schedule y^* in Y and the total transfer is the same as that of the full participation case.*

Proof: If the total transfer of the agents in I is zero, i.e., $\sum_{i \in I} \delta_i^* = 0$, then under Groves-Clarke mechanism, $\delta_i = 0^* \forall i \in I$. It is trivial to show that the proposition holds if there is only one agent in I since this corresponds to the case of a non-pivotal agent whose participation affects neither the selection of an optimal schedule or the total transfer. For more general cases where $|I| > 1$, denote the set of participating (and truthfully reporting) agents by I' . Let $y^*(\mu_{I'} = \pi_{I'})$ denote the schedule that is selected in this partial participation case. Since any agent $i \in I$ is non-pivotal under

5.2. THE SCHEDULE SELECTION GAME

the full participation case, for every $i \in I$, we have

$$\sum_{i' \neq i} \pi_{i'}(y^*(\mu = \pi)) = \sum_{i' \neq i} \pi_{i'}(y^*(\mu_{-i} = \pi_{-i}))$$

or more precisely, $y^*(\mu = \pi) = y^*(\mu_{-i} = \pi_{-i}) \forall i \in I$. Moreover, for every $i \in I$,

$$\sum_{i' \neq i} \pi_{i'}(y^*(\mu = \pi)) = \sum_{i' \neq i} \pi_{i'}(y^*(\mu_{-i} = \pi_{-i})) \geq \sum_{i' \neq i} \pi_{i'}(y) \forall y \in Y,$$

or,

$$\sum_{i' \in I'} \pi_{i'}(y^*(\mu_{I'} = \pi_{I'})) \geq \sum_{i' \in I'} \pi_{i'}(y) \forall y \in Y$$

This means that each agent individually does not affect the selection of an optimal schedule, i.e., without agent i , $i \in I$, the others in I' choose $y^*(\mu = \pi)$, and this in turn leads to the agents in I' collectively selecting $y^*(\mu_{I'} = \pi_{I'}) = y^*(\mu = \pi)$. Because the set I is overall nonpivotal, their total transfer is still 0, hence the overall total transfer does not change. $\diamond$

This shows that to make an optimal schedule selection, participation from the nonpivotal agents is not essential, i.e., full participation is not necessary for optimal schedule selection. However, if any of the pivotal agents choose not to participate then the schedule selected without their messages may be suboptimal, i.e., $y^*(\mu_{I'} = \pi_{I'}) \neq y^*(\mu = \pi)$ when non-participating agents I include pivotal agents. Hence, for the full “utilization” of the mechanism (to select the best schedule in Y), we need to guarantee participation from the pivotal agents. In general, as mentioned above, the mechanism designer needs to consider *individual rationality* or *participation* constraints to have (all) the agents participate the process. Normally, these constraints take the form of

$$u_i(y, \delta_i) \geq u_i(\bar{y}, \bar{\delta}_i)$$

where pair $(\bar{y}, \bar{\delta}_i)$ denote the outcome that would result in case the agent i does not participate in the mechanism, or more generally, that would result without any coordination among the agents. This means that, for every agent, the mechanism should guarantee a positive gain from participation with respect to “uncooperative”

5.3. AN INDUSTRY CASE AND ILLUSTRATIVE EXAMPLE

solutions. Thus, to have all the agents participate in the process, the mechanism needs to compensate for the losses of the agents who lose from cooperating with respect to the “noncooperating” case. Since we cannot discriminate against agents based on the fact that they would have zero transfer in the full participation case, we cannot consider the participation constraints only for those agents, i.e., the participation constraints must be considered for all the agents.

Note that the exact form of the individual rationality constraints depends on the outcome that would be selected in an “uncooperative” case that defines outcome $(\bar{y}, \bar{\delta}_i)$. Additional complexity comes from the fact that there are different levels of implementation of the individual rationality constraints, such as *ex ante*, and *ex post* individual rationality [59]. In general, even satisfying the less demanding *ex ante* individual rationality conditions using a Groves mechanism is not trivial. This negative result is magnified when the agents have a large negative impact on other agents’ profits as in our case: Job agents compete for resources to finish early, and in general, a low tardiness for a job might mean a high tardiness for another. Thus, an effect of cooperation might be to choose a schedule that significantly reduces an agent’s (before transfer) profits from what the agent could achieve when there were no cooperation. In this case, if the participation is voluntary, then the agent would not have an incentive to participate the mechanism. One possible solution of this difficulty is to require that the agents have positive benefits from cooperation in terms of long-term expectation. This means that the agents will gain from participation in the long run over repeated runs of the mechanism. See [35] for an example. We will leave this subject for future study.

5.3 An Industry Case and Illustrative Example

5.3.1 Background

In this section, we use an industry case study to illustrate how the schedule selection game outlined in the previous section can be implemented in a real world environment. We base our example on real scenarios and data that we collected from an

5.3. AN INDUSTRY CASE AND ILLUSTRATIVE EXAMPLE

electronics assembly plant. The plant produces electronic components for automotive manufacturers worldwide. These components include electronic engine control (EEC) modules, anti-lock brake systems (ABS) control, speed control unit and others. A variety of products are produced in this facility. There are *product managers* each responsible for the production of a family of similar products. By nature of the automotive supply chain, each manager has his/her own product specific constraints and preferences driven by contractual agreements with each customer and supplier, delivery arrangements, demand characteristics, quality and technological specifications, just to name a few. The product managers must compete for the same set of resources while trying to maximize his/her own interests and preferences at the moment. The shared resources are a collection of automated production lines called the consolidated surface mount device (SMD) area. Approximately 80% of all electronic products in the plant go through this area, where a pick and place process using a series of SMD machines and testing stations form the main operations. The focus has been on maximizing the efficiency of the capital intensive SMD area while aligning plant-wide objectives with product level requirements so as to maintain responsiveness to customers.

Ideally, each agent in our distributed environment would correspond to a different product manager who is a decision making unit in the actual system. However, due to the way scheduling data is currently managed, we collect primarily order data from the EEC product family, where a single product manager and several schedulers are responsible for the production. While the case example does not capture the real competition among different product managers, it does simulate the distinctly different interests of customers and their products. We assume that each job agent represents an order or a *lot* with a determined number of specific type from the EEC family. In constructing the case, it quickly became apparent that designing an “agent decision structure” is a non-trivial task of its own. Issues such as how to designate the right level of responsibilities to the right agent (e.g., how many products an agent should represent, in what level of details, etc.) maybe critical to the overall performance of the decision structure. Here, we assume a particular design which is consistent with our initial view where each job agent is responsible for

5.3. AN INDUSTRY CASE AND ILLUSTRATIVE EXAMPLE

exactly one job/order. An implicit assumption is that a human production manager responsible for multiple products would have access to multiple job (software) agent's utilities. But the mechanism does not consider the potential dependency among these products. This particular design can be extended to a more complex agent design structure where each agent is responsible for a multiple of orders for a product type. However, the product manager may use a more complex utility function to sort out the priority trade-off among his/her products. This does not change the basic structure of the schedule selection game.

As the production manager clearly pointed out to us, the main bottleneck in the production environment is the SMD resources and operations. This allows us to assume that a reasonable lead time with low variation can be approximated for downstream, non-bottleneck operations. Therefore, we may view scheduling as a single-stage problem where each job consists of a single operation with its due-date calculated by subtracting the downstream lead time from the original order due-date. Although the problem is single-stage, an SMD operation can be performed by any machine in a specific subset of the SMD lines. There are currently around 10 SMD machines on the line and one or more of these (up to 5) can serve as alternatives for a specific SMD operation. However, for a given service mount operation these alternative machines are not identical in processing speed, yield, and quality.

To illustrate the significance of private information and local constraints a job agent maintains, the issues surrounding order due-dates provide good examples. Order due-dates are at best moving targets subject to frequent revisions from the customer (via electronic schedule releases, or informal off-line communications). Worse, due-dates are often coupled with business issues such as pricing, potential discounts, consolidation of freights due to transportation or container restrictions, future order status, long-term good will, and inventory restrictions, just to name a few. In other words, behind a simple due-date performance measure there is a significant amount of private information only known to the product manager in charge. As one of the product managers stated, the managers may spend more than half of his time dealing with these customer related issues. This does not even include other product specific issues related to component suppliers, transportation providers, or the policies and

5.3. AN INDUSTRY CASE AND ILLUSTRATIVE EXAMPLE

reward systems specific to the facility, all of which influence a product manager's decision.

5.3.2 A Numerical Example

In the following, we construct a numerical example to illustrate some of the finer points of the schedule selection game. The numbers used in the example are from real data. We make the following implementation decisions:

- We assume that a primary concern of each job agent is to meet job due-dates.
- We assume that lot sizing and routing decisions are made before scheduling decisions. This assumption leads to two important simplifications: (1) setup times can be added to the processing times of the operations, and (2) the problem can be decomposed into a series of single machine problems since the machine assignments are made through routing decisions. We use the lot-sizing and routing decisions made by the scheduling software that has been implemented in the plant during the test period. Note, however, any algorithm can be used instead. Hence, we can think of the scheduling process as an integral part of a larger planning/routing/scheduling decision structure where the higher-level decisions are an input for our scheduling process.
- At any one time there may be thousands of jobs in the system database. To limit ourselves to a reasonable number of jobs, we constructed a “candidate job set” which have due dates smaller than a certain threshold value. In this way, we eliminated jobs that would not be scheduled in the current planning horizon even if they were considered. In the following numerical example, we excluded the jobs (i.e., do not put them in candidate job set) with due dates later than 10 days from current date.
- We assume *hourly* time buckets for our discrete time model of scheduling problem. Due to this, we needed to consolidate really short jobs to form “super” jobs that have sizes comparable to our time bucket. We use the same planning horizon (5 days, 120 hours) that is employed in practice.

5.3. AN INDUSTRY CASE AND ILLUSTRATIVE EXAMPLE

Formally, each job agent i , and his job, can be represented by a 4-tuple (a_i, m_i, p_i, d_i) , where a_i is the release time (or earliest start time) of job i , m_i is the SMD machine that is assigned, p_i is the processing time requirement on machine m_i including setup time, and d_i is the derived job due date.

To construct the candidate schedules in set Y , we used the Lagrangean-based auction-theoretic algorithm developed in our earlier studies [46, 49]. We use two ways to generate the set of feasible schedules for selection: (1) schedules generated by applying feasibility restoration to a given set of Lagrangean rankings result from the auction, and (2) schedules generated by applying variable fixing in 1 out of K ranking constraints result from our “proactive equilibrium” study. Thus, we have a set of feasible schedules from which we form the schedule selection game.

We view the implementation of a direct revelation mechanism, or more specifically a Groves-Clarke mechanism, as follows: All candidate schedules in set Y are posted on a public (maybe electronic) bulletin board. The job agents (maybe software as well) then provide their own valuation for each schedule using their own criteria based on private information. Since each agent’s performance depends only on its job’s allocation in a schedule and its transfer, an agent does not need to see the allocations of other jobs. This procedure is therefore asynchronous. When all job agents provide the necessary valuation, the schedule with the maximum total reported utility is selected as an outcome of the mechanism and the transfers are calculated. To simulate this process, we use the negative value of weighted tardiness for all utility measures.

To further illustrate this step, we assume that there is a “utility” evaluation table to be filled by the job agents. Each job agent has access to its part of each schedule and fills a value for each schedule in a space reserved for it. See Table 5.1 for an example of such a table. In this example, we have 33 jobs to be scheduled on a single SMD machine, CSMD10C. Note that similar tables can be constructed for other jobs and resources since they are independent. We only present one of them here. In the more complex job shop scheduling problem, we would still have a single table for relating the jobs to each schedule. To have a manageable size example, we have created 8 schedules. Each job agent in this example fill the tardiness values for each

5.3. AN INDUSTRY CASE AND ILLUSTRATIVE EXAMPLE

of these 8 schedules. The table shows the (truly) reported tardiness values ($\mu_i^*(y)$) for each job under each schedule.

Using these reported values, the Groves-Clarke mechanism selects a schedule using the schedule selection function $y^*(\mu^*)$ (since our goal is to maximize total utility or *minimize* total weighted tardiness, we use a minimization version of schedule selection function). Thus, the selected schedule (y^*) is the last one (schedule 8) in the table with a total tardiness of 185. The mechanism next calculates the transfers from agents. Recall that, to calculate Clarke transfer for agent i , we need to find what the other agents would select without i 's messages, $y_{-i}^*(\mu_{-i})$. The column titled y_{-i}^* lists the schedules that would be selected if the corresponding agent did not report any values. Note that if an agent does not affect the others' selection (i.e., $y_{-i}^* = y^* = 8$) then his transfer is zero, otherwise he pays the increase that he makes on others' total tardiness. Consider, for example, job number 6. Without job 6's reported values, the others would select schedule 1 with total tardiness of 172 ($= \min_{y \in Y} \sum_{i' \neq 6} \mu_{i'}(y_{-6}^*(\mu_{-6}))$). The difference between the tardiness of schedule 8 and that of 1 can be seen due to the involvement of job 6. Thus, the agent for job 6 pays the difference: $185 - 172 = 13$. Note that even after paying this amount, the agent's total cost (tardiness + payment) with the selection of schedule 8 is 13 ($= 0 + 13$) which is still lower than what would be if the other agents collectively select schedule 1 ($13 < 29$) without its participation. We can show that this is true for every job agent. The agents with negative transfers (1, 4, 6, 8, 15, 19, 26, 27, 30, and 31) are charged 129 units in total, i.e., they pay this amount to collectively select schedule 8 instead of another alternative. The last column in the table lists the agents' total costs after transfer. Finally, note that, although each agent individually has a positive benefit with respect to what the others would choose, it may be worse off with respect to another schedule. For example, agent 6 pays 13 (with total cost of 13) but there are in fact schedules with tardiness values of 8 or even 0 (assuming these are without transfers, recall individual rationality conditions). In fact, some agents (including 6) are worse off even when their total costs are compared to the expected tardiness for which a schedule is randomly (with equal probability across schedules) selected from set Y .

5.3. AN INDUSTRY CASE AND ILLUSTRATIVE EXAMPLE

Table 5.1: An example table for the Groves-Clarke mechanism for 8 schedules of 33 jobs

Job (i)	Schedules ($y \in Y$)								y_{-i}^*	$-\delta_i^*$	$\pi_i - \delta_i^*$
	1	2	3	4	5	6	7	8			
1	36	4	4	4	4	0	0	0	1	20	20
2	7	6	11	9	9	0	0	0	8	0	0
3	4	11	0	0	0	21	0	4	8	0	4
4	0	14	0	0	0	0	15	0	7	9	9
5	0	0	0	0	0	0	0	22	8	0	22
6	29	0	0	8	8	0	0	0	1	13	13
7	8	19	24	10	10	0	0	0	8	0	0
8	0	16	0	0	65	0	17	0	7	11	11
9	16	0	5	12	12	0	0	0	1	0	0
10	0	2	7	14	14	0	0	0	8	0	0
11	0	25	30	14	18	0	0	0	8	0	0
12	0	27	32	16	15	0	0	0	8	0	0
13	0	6	11	16	16	0	0	0	8	0	0
14	0	6	11	16	16	0	0	23	8	0	23
15	0	14	0	0	0	21	15	0	7	9	9
16	0	7	12	17	17	0	0	24	8	0	24
17	26	21	1	21	0	63	68	67	8	0	67
18	0	32	37	17	16	0	0	0	8	0	0
19	29	37	8	8	8	0	0	0	1	13	13
20	0	36	38	18	17	0	0	0	8	0	0
21	11	39	41	13	13	0	0	0	8	0	0
22	0	17	19	20	20	20	0	0	8	0	0
23	0	0	0	0	0	0	5	5	8	0	5
24	0	18	20	21	21	21	0	24	8	0	24
25	0	0	0	0	0	0	4	4	8	0	4
26	0	18	0	0	67	0	19	0	7	13	13
27	0	20	0	0	69	0	21	0	7	15	15
28	14	48	10	16	16	0	0	0	8	0	0
29	0	21	1	21	70	0	0	0	8	0	0
30	1	25	25	26	26	26	0	0	6	5	5
31	8	29	29	30	30	30	27	0	7	21	21
32	0	52	52	19	18	0	0	0	8	0	0
33	12	0	61	11	54	4	0	12	8	0	12
Total	201	570	489	377	649	206	191	185		129	314

5.4 Discussion

We now discuss several interesting issues related to incentive compatible schedule selection in light of the illustrative example.

5.4.1 Generation of Schedules

A direct revelation mechanism for distributed scheduling involves many inter-related issues in order for it to be implemented successfully: First of all, we need to address the question of *how* the schedules collected in set Y are generated. Obviously schedules in this set affect the quality of the final schedule selected in a direct revelation mechanism. It is also important since it affects the computational implication of the overall mechanism.

We first assume that each schedule in Y represents a feasible time slot allocation from the shared resources to the job agents, i.e., in each schedule, non-overlapping contiguous blocks of time slots from each machine are assigned to the job agents for their respective operation on that machine. We may interpret the set of the candidate schedules as a collection of Gantt charts. Thus, we can identify a schedule y in set Y by using job - machine - time slot assignment indicators such as $X_{ikt}(y) = 1$ if job i is assigned to time slot t on machine k in schedule y . In this representation, a job agent does not need to know the full schedule (related to the others) to compare the schedules; knowing its own assignment is enough to compute its net utility. A job agent can compute the value of a schedule (profit that it can make) by using the part of the schedule that is related to its job. If we collect all the time slots assigned to agent i in $X_i(y) \equiv \{X_{ikt}(y)\}_{k=1,\dots,M,t=1,\dots,T}$, then the job agent can compute its profit by using X_i :

$$\pi_i(y) = \pi_i(X_i(y))$$

We know that the number of schedules to be presented to the agents are limited by memory and computer time. An important issue is which schedules for the original scheduling problem should be put in the set of “candidate schedules”. An important result regarding the quality of the final schedule collectively selected is that if the set

5.4. DISCUSSION

of candidate schedules Y consists of an optimal schedule of the original scheduling problem, say JSP, then this schedule will definitely be selected through the mechanism, i.e., if Y is guaranteed to have a certain performance guarantee on the total profit achievable, then the schedule selected using the schedule selection function has the same performance guarantee. This is summarized in the following proposition.

Proposition 5.6 *The Groves mechanism is guaranteed to choose a global optimal schedule of the original problem and achieve the optimal joint profit if the set Y consists of an optimal schedule, i.e., $\sum_i \pi_i(y^*) = v(\text{JSP})$, where $v(\text{JSP})$ denotes the value of a (global) optimal schedule.*

Then the remaining question is how do we select a subset of good schedules for set Y ? What if we are interested in including the optimal schedule in this set? We list several alternatives for the construction of the candidate schedule set as follows:

- *Historic information:* In most practical environments, schedule selection is not a one time process, but an exercise to be repeated time and again. If this is the case, an important source of information for the generation of quality schedules is the performance of those selected in the past. Schedules that have been implemented in the past, with their realized performance measures, may help identify promising schedules for the current selection process. Of course the usefulness of the historic information depends on the variability of problem instances. Suppose that every week a similar set of jobs (similar routing, processing, and release and shipment requirements) compete for shared machines. It is likely that the past performance might reveal patterns in the schedules that can be used to generate candidate schedules.
- *Preprocessing:* We can view the schedule selection game as a stage in a distributed scheduling procedure preceeded by a preprocessing or screening stage. In this view, a set of high-quality schedules are collected in set Y in the preprocessing stage. The two methods outlined in our previous works [46, 49] will be useful to isolate good quality candidate schedules. These are in fact what we

5.4. DISCUSSION

used to generate the candidate schedules in the example. The feasibility restoration routine applied to a Lagrangean schedule as outlined in [46] will produce a number of schedules which can be put in set Y . In [49], we showed that one of a small number of precedence (or ranking) constraints must hold in an optimal solution. A preprocessing stage will involve fixing one out of these alternative precedence constraints among the jobs and generating different schedules each of which satisfies an alternative ranking. Since, we know that this preprocessing does not eliminate the optimal schedule, the schedule selection mechanism will be able to select the optimal schedule for the original problem. The only prohibitive element of this type of preprocessing is the number of the candidate schedules allowed, that is, the size of Y . This may be controlled to a certain degree by effectively excluding some of the rankings from consideration although a fully optimal schedule cannot be guaranteed in this case.

Another rather different alternative is to have the agents propose full feasible schedules that are supposed to improve a default schedule that will be implemented in case any better schedule is not proposed for a certain time period. One such implementation for train scheduling can be found in [10] where the agents are provided with the incentives, i.e., rebates from the global performance improvement, to propose high-quality schedules. Note that, in this case, although the generation of schedules are carried out along with the online selection of a schedule, the private information assumption is rather destroyed since each agent is assumed to have full information to propose a fully feasible schedule. Note that these methods are not mutually exclusive, i.e., they can be combined in a single process to make the candidate schedule set smaller and more effective in finding an optimal schedule.

5.4.2 Aligning Original Performance and Transfers

As we have outlined above, an issue is to align the “normal” performance measure (e.g., weighted tardiness) with the unit of transfers. This is a prerequisite for a sensible “utility” definition of the agents, i.e., after transfer (net) performance, $u_i(y) = \pi_i(y) + \delta_i$. The agents should take into account the effects of transfers as well as the

5.4. DISCUSSION

initial performance. In the mechanism design literature, both the initial performance and the transfers are interpreted and expressed in monetary terms, which is what we did in our discussion. However, when the two are not in the same units, we need a normalizing function to convert the initial performance such that the overall utility makes sense for the agents. For example, one can use a function that calculates actual tardiness costs including any late charges or express shipments costs. Thus, the “profit” of agent i is defined as

$$\pi_i(y) = f(T_i),$$

where T_i is tardiness of job i and f is a conversion function.

Furthermore, the transfers should be explicitly processed through actual transactions so that the transfers can affect overall utility of the agents. Considering that actual monetary transactions can be difficult to implement in a intra-plant competition, we may imitate the environment by distributing some type of “currency” to the agents and allow them to spend for better scheduling selections. Another alternative is to align the incentives of the agents with their overall performance using an “evaluation measure”. In this way, the compensation to an agent will only and directly depend on the the evaluation measure, which is the net utility of the agent after transfers. Note that in the final interpretation, we assume the existence of a controller or a “center” that rewards or punishes the agents depending on their realized evaluation measures.

5.4.3 Coordinated Scheduling with a Central Authority

We have presented the mechanism in the previous sections as a collective decision making process without a center. However, it may be modified to accommodate a central authority. For example, if we assume that (1) the center is responsible for generating the schedules in set Y , (2) selects a mechanism with a message space and an outcome function that chooses a schedule and arranges the transfers, (3) collects money from the agents with negative δ_i and pays to the agents with positive δ_i , then there are only technical differences which are not critical for our analysis.

5.4. DISCUSSION

However, if we consider a central authority who makes decisions that may influence the agents' decisions and who has authority over the agents' actions, the issues change. The center here may allocate resources (e.g., time windows on machines assigned to job agents) that limit the decisions left to the local agents, and he/she may choose to reward or punish the agents depending on their actual performance (note that in the extreme case, the center's decisions may induce exactly one solution in the agent's feasible space). The main difficulty for an "authoritarian center" while making these decisions is that he/she may not have perfect information about local constraints and preferences so that sound decisions may not be possible, or alternatively, it may be essential to elicit private information from the agents in order to maximize the center's utility. Since the overall performance is the result of decisions from both the center and the agents, the nature of the coordination and the balance of the authority present a whole different design problem.

We note the following differences as compared to the centerless design:

- The main issue here is to elicit private information from the local agents so that (1) the center makes the best allocation of shared resources, and then (2) the agents make best local decisions that are aligned with the global objectives.
- The center might have different alternatives to provide to the agents with the incentives that are compatible with the overall objective. One common tool is an evaluation measure for the agents that has been used in other Groves mechanisms (see, e.g., [37]). When the compensation to an agent is tied to the evaluation measure of the agent, the incentive compatibility is assured. As expected, the form of the evaluation measure function is similar to the net utility function defined in previous sections.
- For this mechanism to be implemented, we need to assume that (1) the center can enforce resource assignments but he has no control on the local decisions of the agents, and (2) the center can measure the final performance of the agents at the end of the execution of a schedule (realized values of profits) and reward or punish them according to these realized performances. Hence, the center is assumed to have a role of both a decision maker and a *controller*.

5.5. CONCLUSIONS

Thus, the mechanism is not only a direct revelation mechanism from a game theoretic point of view, but also a *control mechanism* from an organization control view [37].

- Possible steps for such mechanism may be as follows: (1) The agents are asked to report *maximum profit* that they can achieve for possible time window assignments. (2) The center makes time window assignment decisions based on the reported values by the agents. (3) The agents make the local decisions under the center's time window decisions, and they use their assigned time windows to process their jobs. (4) At the end of schedule execution, each agent's performance is measured, and rewards are calculated and distributed to the agents.

5.5 Conclusions

In this study, we address the issue of incentive compatibility in an environment we called a “schedule selection game.” We showed that despite the difficulties involved in balanced budget and individual rationality, an incentive compatible mechanism can be devised for distributed scheduling. We also discuss additional challenges and issues posed by the scheduling game, such as schedule generation, and transfer units. We explore the application context of this scheduling game using an industry case and we illustrate the mechanism using a numerical example drawn from a real-life setting. Although we touched on topics such as alternative schedule generation, asynchronous scheduling, and other agent designs, in-depth coverage of these topics are left for future study.

Chapter 6

Conclusions

6.1 Main Contributions

As we mentioned in previous chapters, we felt that there is a need for distributed decision making models and algorithms for scheduling in today's distributed manufacturing environment. Centralized or hierarchical approaches based on monolithic models do not fill this need satisfactorily because of their inherent properties (e.g. time-dependent local objectives, preferences, and constraints cannot be considered in a monolithic centralized model very effectively). One of the contributions of this research is to satisfy this need by proposing a new distributed decision structure.

In this chapter we summarize the main contributions of our research:

- We believe the main contribution of Chapter 2 is the structural mapping and integration of models and approaches from auction theory and operations research. We propose four price-based auction mechanisms (using protocols and payment functions) in the context of distributed scheduling and we uncovered the relationship between auction mechanisms and Lagrangean-based decomposition of the job shop scheduling problem. We examine different means of decomposition and their corresponding auction mechanisms for scheduling problems. The study is a first step towards a more thorough investigation of this emerging area. The chapter also contributes in terms of aligning the local interests of agents with the global objectives by modifying their utilities through

6.1. MAIN CONTRIBUTIONS

carefully designed payment functions. We first revealed that the main issues in distributed mechanisms are solution feasibility, convergence and communication overhead, and information availability. This lead us to intensify our efforts on these issues in our later work. This chapter also provides a rather unique and unifying review of the relevant literature from operations research, computer science and economics.

- The main contribution of Chapter 3 is to demonstrate that both stochastic analysis and dynamic adaptation could significantly improve scheduling robustness. Making use of second moment stochastic information achieves significant improvement in a majority of the cases. More importantly, the benefit of stochastic analysis are more significant in larger size problems at a higher level of uncertainty. We also showed that the marginal benefit of stochastic preprocessing is also affected by the assumed statistical distributions and problem instances. This chapter also illustrates how a Lagrangean-based approach which provides a lower bound and infeasible solution can be used for heuristic solution generation.
- Chapter 4 contributes mainly by outlining the theoretical requirements of a distributed approach which tries to consolidate an optimal solution by seeking an equilibrium among the agents. From this perspective, we outlined an equilibrium condition that unifies feasibility and optimality. We also partially answered how the issues of convergence and communication overhead can be overcome by exploiting the special structure of the problem
- In Chapter 5, our main contribution was to provide insights on the issues of incentive compatibility and truthful information revelation which were overlooked in previous OR based studies. This chapter also contributes by showing another way of aligning local interests with global objectives through transfer mechanisms. We also implemented a seemingly unrelated tool from economics, Groves-Clarke mechanism, to our scheduling environment.

We next summarize overall contributions, not pertinent to any specific chapter:

6.1. MAIN CONTRIBUTIONS

- Our overall research provides insights in resolving peer-to-peer conflicts among decision making agents who have conflicting interest due to shared resources and private information.
- Our research provides important insights in the correspondence between system optimality and the price equilibrium of individual entities in the system. These insights help build a basis for distributed implementation of monolithic optimization models.
- Our research establishes a pathway which allows us to study distributed decision making using the structure of traditional optimization. We take advantage of the wealth of mathematical explorations and computational methods in optimization for efficient implementation of distributed decision making.
- A single-point optimization view may impose solutions that are not equitable in any measure for any agent(s). This implies that the state implied by the solution may not be stable or in equilibrium. In this research, we attempt to devise compensation/transfer mechanisms so that some sort of “stability” in the solution is achieved.
- We address the communication and feasibility issues in distributed environment, which are normally satisfied by a monolithic optimization model if used. We analyze both the amount of communication and the interference by an outside authority (or a center) for feasibility restoration.
- We finally examine practical issues involved in implementing distributed decision making models and algorithms using real industry data and scenarios.

For future research directions, please see the conclusions of each chapter.

Bibliography

- [1] A. D. Baker. Metaphor or reality: A case study where agents bid with actual costs to schedule a factory. In S. H. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, River Edge, NJ, 1996.
- [2] J. S. Banks, J. O. Ledyard, and D. P. Porter. Allocating uncertain and unresponsive resources. *RAND Journal of Economics*, 20(1):1–25, 1989.
- [3] P. Baptiste and J. Favrel. Taking into account the rescheduling problem during the scheduling phase. *Production Planning and Control*, 4:349–360, 1993.
- [4] J. C. Bean, J. R. Birge, J. Mittenenthal, and C. E. Noon. Matchup scheduling with multiple resources, release dates and disruptions. *Operations Research*, 39(3):470–483, 1991.
- [5] D. P. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14:105–123, 1988.
- [6] D. P. Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. *Interfaces*, 20(4):133–149, 1990.
- [7] D. P. Bertsekas. *Linear Network Optimization*. MIT Press, Cambridge, MA, 1991.
- [8] D. P. Bertsekas. Auction algorithms for network flow problems. *Computational Optimization and Applications*, 1:7–66, 1992.

BIBLIOGRAPHY

- [9] S. Bikhchandani and J. W. Mamer. Competitive equilibrium in an exchange economy with indivisibilities. *Journal of Economic Theory*, 74:385–413, 1997.
- [10] P. J. Brewer. Decentralized computation procurement and computational robustness in a smart market. Technical report, Department of Economics, Georgia State University, Atlanta, GA, 1997.
- [11] P. J. Brewer and C. R. Plott. A binary conflict ascending price (BICAP) mechanism for the decentralized allocation of the right to use railroad tracks. *International Journal of Industrial Organization*, 14:857–886, 1996.
- [12] E. S. Byeon, S. D. Wu, and R. H. Storer. Decomposition heuristics for robust job shop scheduling. Technical Report 93T-008, Lehigh University, Bethlehem, PA, 1993, to appear in *IEEE Transactions on Robotics and Automation*.
- [13] J. Carlier and E. Pinson. Adjustments of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78:146–161, 1994.
- [14] A. Charnes and W. W. Cooper. Chance-constrained programming. *Management Science*, 6:73–79, 1959.
- [15] J. W. Chinneck. Feasibility and viability. In T. Gal and H. J. Greenberg, editors, *Advances in Sensitivity Analysis and Parametric Programming*. Kluwer Academic Publishers, Boston, MA, 1997.
- [16] J. W. Chinneck. Finding a useful subset of constraints for analysis in an infeasible linear program. *INFORMS Journal on Computing*, 9(2):164–174, 1997.
- [17] L. K. Church and R. Uzsoy. Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, 5:153–163, 1992.
- [18] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.
- [19] S. I. Cohen. Incentives, iterative communication, and organizational control. *Journal of Economic Theory*, 22:37–55, 1980.

BIBLIOGRAPHY

- [20] S. I. Cohen. Truth-telling, dominant strategies, and iterative Groves mechanisms. *Public Choice*, 51:333–343, 1986.
- [21] C. J. Corbett, F. J. C. Debets, and L. N. Van Wassenhove. Decentralization of responsibility for site decontamination projects: A budget allocation approach. *European Journal of Operational Research*, 86:103–119, 1995.
- [22] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983.
- [23] G. Demange, D. Gale, and M. Sotomayor. Multi-item auctions. *Journal of Political Economy*, 94(4):863–872, 1986.
- [24] R. Engelbrecht-Wiggans. Auctions and bidding models: A survey. *Management Science*, 26(2):119–142, 1980.
- [25] R. Engelbrecht-Wiggans. An introduction to the theory of bidding for a single object. In R. Engelbrecht-Wiggans, M. Shubik, and R. M. Stark, editors, *Auctions, Bidding, and Contracting: Uses and Theory*, pages 53–103. New York University Press, New York, NY, 1983.
- [26] M. L. Fisher. An application oriented guide to Lagrangian relaxation. *Interfaces*, 15(2):10–21, 1985.
- [27] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Boyd and Fraser Publishing Company, Danvers, MA, 1993.
- [28] J. W. Friedman. *Game theory with applications to economics*. Oxford University Press, New York, NY, 1990.
- [29] G. Gallego. Linear control policies for scheduling a single facility after an initial disruption. Technical Report 770, Cornell University, Ithaca, NY, 1988.
- [30] G. Gallego. Produce-up-to policies for scheduling a single facility after an initial disruption. Technical Report 771, Cornell University, Ithaca, NY, 1988.

BIBLIOGRAPHY

- [31] R. L. Graves, L. Schrage, and J. K. Sankaran. An auction method for course registration. *Interfaces*, 23(5):81–92, 1993.
- [32] J. Green and J. J. Laffont. Characterization of strongly individually incentive compatible mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.
- [33] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.
- [34] T. Groves. Incentive compatible control of decentralized organizations. In Y. C. Ho and S. K. Mitter, editors, *Directions in Large-Scale Systems: Many-person Optimization and Decentralized Control*, pages 149–185. Plenum Press, New York, NY, 1976.
- [35] T. Groves. Information, incentives and the internalization of production externalities. In S. A. Lin, editor, *Theory and Measurement of Economic Externalities*, pages 65–83. Academic Press, New York, NY, 1976.
- [36] T. Groves. Efficient collective choice when compensation is possible. *Review of Economic Studies*, 46:227–241, 1979.
- [37] T. Groves and M. Loeb. Incentives in a divisionalized firm. *Management Science*, 25(3):221–231, 1979.
- [38] F. Gul and E. Stacchetti. English and double auctions with differentiated commodities. Technical Report 97-02, University of Michigan, 1997.
- [39] F. Gul and E. Stacchetti. Walrasian equilibrium without complementarities. Technical Report 97-01, University of Michigan, 1997.
- [40] A. Hylland and R. Zeckhauser. An efficient allocation of individuals to positions. *Journal of Political Economy*, 87(2):293–314, 1979.
- [41] T. Ichiiishi. *Game theory for Economic Analysis*. Academic Press, New York, NY, 1983.

BIBLIOGRAPHY

- [42] P. Jennergren. A price schedules decomposition algorithm for linear programming problems. *Econometrica*, 41(5):965–979, 1973.
- [43] R. A. Jose, P. T. Harker, and L. H. Ungar. Auction and optimization: Methods for closing the gap caused by discontinuities in demands. Technical report, University of Pennsylvania, 1997.
- [44] E. Kutanoglu and I. Sabuncuoglu. Experimental investigation of scheduling rules in a dynamic job shop with weighted tardiness costs. In *Proceedings of 3rd Industrial Engineering Research Conference*, Atlanta, GA, 1994.
- [45] E. Kutanoglu and I. Sabuncuoglu. An analysis of heuristics in a dynamic job shop with weighted tardiness objectives. *International Journal of Production Research*, 37(1):165–187, 1999.
- [46] E. Kutanoglu and S. D. Wu. On combinatorial auction and Lagrangean relaxation for distributed resource scheduling. Technical Report 97T–12, Department of Industrial and Manufacturing Systems Engineering, Lehigh University, 1997, To Appear in IIE Transactions, Special Issue on Game Theory Applications in Industry.
- [47] E. Kutanoglu and S. D. Wu. Improving schedule robustness via stochastic analysis and dynamic adaptation. Technical Report 98T–01, Department of Industrial and Manufacturing Systems Engineering, Lehigh University, 1998.
- [48] E. Kutanoglu and S. D. Wu. Incentive compatible scheduling. Technical Report 99T, Department of Industrial and Manufacturing Systems Engineering, Lehigh University, 1999.
- [49] E. Kutanoglu and S. D. Wu. Proactive equilibrium for distributed job shop scheduling. Technical Report 99T, Department of Industrial and Manufacturing Systems Engineering, Lehigh University, 1999.
- [50] B. Lageweg, J. K. Lenstra, and A. H. G. Rinnooy Kan. Job shop scheduling by implicit enumeration. *Management Science*, 24:441–450, 1977.

BIBLIOGRAPHY

- [51] H. B. Leonard. Elicitation of honest preferences for the assignment of individuals to positions. *Journal of Political Economy*, 91(3):461–479, 1983.
- [52] J. Liu and K. P. Sycara. Distributed constraint satisfaction through constraint partition and coordinated reaction. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, 1993.
- [53] J. Liu and K. P. Sycara. Distributed problem solving through coordination in a society of agents. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, 1994.
- [54] J. Liu and K. P. Sycara. Exploiting problem structure for distributed constraint optimization. In *Proceedings of the 1st International Conference on Multi-agent Systems*, San Francisco, CA, 1995.
- [55] J. Ma. English auctions and walrasian equilibria with multiple objects: A dynamic approach. Technical Report 97-02, Rutgers University, 1997.
- [56] J. Ma. Competitive equilibrium with indivisibilities. Technical Report 98-09, Rutgers University, 1998.
- [57] M. J. Maddox and J. R. Birge. Using second moment information in stochastic scheduling. In G. Yin and Q. Zhang, editors, *Recent Advances in Control and Optimization of Manufacturing Systems*, pages 99–120. Springer-Verlag, 1992.
- [58] T. W. Malone and S. A. Smith. Modeling the performance of organizational structures. *Operations Research*, 36(3):421–436, 1988.
- [59] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, New York, NY, 1995.
- [60] R. P. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25(2):699–738, 1987.
- [61] S. V. Mehta and R. Uzsoy. Predictable scheduling of a job-shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, Forthcoming:149–156, 1997.

BIBLIOGRAPHY

- [62] P. R. Milgrom and R. J. Weber. A theory of auctions and competitive bidding. *Econometrica*, 50(5):1089–1122, 1982.
- [63] T. E. Morton, S. Lawrence, S. Rajagopalan, and S. Kekre. MRP-STAR PATRIARCH’s planning module. *Working Paper, Graduate School of Industrial Administration, Carnegie Mellon University*, 1986.
- [64] T. E. Morton, S. Lawrence, S. Rajagopalan, and S. Kekre. A price-based shop scheduling module. *Working Paper, Graduate School of Industrial Administration, Carnegie Mellon University*, 1988.
- [65] T. E. Morton and D. Pentico. *Heuristic Scheduling Systems with Applications to Production and Project Management*. John Wiley and Sons, New York, NY, 1993.
- [66] A. P. Muhlemann, A. G. Lockett, and C. K. Farn. Job shop scheduling heuristic and frequency of scheduling. *International Journal of Production Research*, 20(2):227–241, 1982.
- [67] D. E. Neiman, D. W. Hildum, V. R. Lesser, and T. W. Sandholm. Exploiting meta-level information in a distributed scheduling system. In *Proceedings of the 12th Conference on Artificial Intelligence*, Seattle, WA, 1994.
- [68] CPLEX Optimization. Using the CPLEX base system including CPLEX mixed integer solver and barrier solver options, 1995.
- [69] R. G. Parker and R. L. Rardin. *Discrete Optimization*. Academic Press, Inc., San Diego, CA, 1988.
- [70] H. V. D. Parunak. Manufacturing experience with the contract net. In M. N. Huhns, editor, *Distributed Artificial Intelligence*, pages 285–310. Morgan Kaufmann Publishers, Los Altos, CA, 1987.
- [71] H. V. D. Parunak. Applications of distributed artificial intelligence. In G. M. P. O’Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, pages 139–164. John Wiley and Sons, 1996.

BIBLIOGRAPHY

- [72] M. Pinedo. *Scheduling: Theory, Algorithms and Systems*. Prentice Hall, Englewood Cliffs, NJ, 1995.
- [73] A. Pritsker, L. Watters, and P. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science: Theory*, 16(1):93–108, 1969.
- [74] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.
- [75] M. H. Rothkopf and R. M. Harstad. Modeling competitive bidding: A critical essay. *Management Science*, 40(3):364–384, 1994.
- [76] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.
- [77] R. O. Roundy, W. L. Maxwell, Y. T. Herer, S. R. Tayur, and A. W. Getzler. A price-directed approach to real-time scheduling of production operations. *IIE Transactions*, 23(2):149–160, 1991.
- [78] N. Sadeh. MICRO-BOSS: A micro-opportunistic factory scheduler. Technical Report CMU-RI-TR-91-22, Carnegie Mellon University, The Robotics Institute, 1991.
- [79] T. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*, pages 11–68. MIT Press, 1987.
- [80] T. Sandholm. An implementation of contract net protocol based on marginal cost calculations. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 256–262, Washington D.C., 1993.
- [81] T. Sandholm. Approaches to winner determination in combinatorial auctions. Technical Report WUCS-99-01, Washington University, Department of Computer Science, 1999, to appear in *Decision Support Systems*.

BIBLIOGRAPHY

- [82] T. Sandholm and V. Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In *14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 694–701, Montreal, Canada, 1995.
- [83] T. Sandholm and F. Ygge. On the gains and losses of speculation in equilibrium markets. In *15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, 1997.
- [84] J. K. Sankaran. On a dynamic auction mechanism for a bilateral assignment problem. *Mathematical Social Sciences*, 28(2):143–150, 1994.
- [85] V. L. Smith. *Papers in Experimental Economics*. Cambridge University Press, New York, NY, 1991.
- [86] R. M. Stark and M. H. Rothkopf. Competitive bidding: A comprehensive bibliography. *Operations Research*, 27:364–391, 1979.
- [87] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constraint heuristic search. *IEEE Transactions on System, Man and Cybernetics*, 21:1446–1461, 1991.
- [88] J.C. Tan and Patrick T. Harker. Designing workflow coordination: centralized versus market-based mechanisms. Technical report, Fishman-Davidson Center, The Wharton School, University of Pennsylvania, 1997, to appear in Information Systems Research.
- [89] D. M. Upton, M. M. Barash, and A. M. Matheson. Architectures and auctions in manufacturing. *International Journal of Computer Integrated Manufacturing*, 4(1):23–33, 1991.
- [90] A. P. J. Vepsalainen and T. E. Morton. Priority rules for job shops with weighted tardiness costs. *Management Science*, 33(8):1035–1047, 1987.
- [91] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

BIBLIOGRAPHY

- [92] K. Wang and D. Veeramani. An adaptive machine bidding strategy for distributed shop-floor control under stochastic part demands. In *5th Industrial Engineering Research Conference Proceedings*, pages 321–326, Minneapolis, MN, 1995.
- [93] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1(1):1–23, 1993.
- [94] M. P. Wellman. Market-oriented programming: Some early lessons. In S. H. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, River Edge, NJ, 1996.
- [95] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Some economics of market-based distributed scheduling. In *Eighteenth International Conference on Distributed Computing Systems*, Amsterdam, The Netherlands, 1998.
- [96] S. D. Wu, E. S. Byeon, and R. H. Storer. A graph-theoretic decomposition of job shop scheduling problems to achieve scheduling robustness. Technical Report 93T-009, Lehigh University, Bethlehem, PA, 1993, to appear in *Operations Research*.
- [97] S. D. Wu, R. H. Storer, and P. C. Chang. A rescheduling procedure for manufacturing systems under random disruptions. In *New Directions for OR in Manufacturing*. Springer-Verlag, 1992.
- [98] S. D. Wu, R. H. Storer, and P. C. Chang. One machine rescheduling heuristics with efficiency and stability as criteria. *Computers and Operations Research*, 20:1–14, 1993.
- [99] M. Yamamoto and S. Y. Nof. Scheduling/rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, 23(4):705–722, 1985.

BIBLIOGRAPHY

- [100] F. Ygge and H. Akkermans. On resource-oriented multi-commodity market computations. In *Proceedings of the 4th International Conference on Multi-Agent Systems ICMAS'98*, 1998.

Vita

Erhan Kutanoglu was born in Amasya, Turkey in 1970. He attended elementary and middle schools in home town, and went to the Science High School in Izmir, Turkey, which is known by its focus on science and mathematics. After graduating from the high school in 1987, he went on to Bilkent University in Ankara, Turkey, for his undergraduate education in Industrial Engineering. He spent his first year at Bilkent by taking English preparation classes. He received his B.S. degree in Industrial Engineering in Spring 1992, and he started his Master's education in the same school in Fall 1992. After receiving his M.S. in Industrial Engineering in Fall 1994, he wanted to pursue his Ph.D. degree in IE/OR in the U.S. and went on to Lehigh University, Bethlehem, PA. He completed his Ph.D. degree requirements in February 1999.

His research interests include large-scale industrial optimization problems, models and algorithms for distributed decision making, game theoretic approaches for combinatorial problems and robustness under uncertainty. The applications include resource allocation, machine scheduling, logistics, and supply chain management problems. He is a member of IIE and INFORMS.