

COORDINATED SUPPLY CHAIN SCHEDULING

Dag Kjenstad

Dr. Ing. Thesis/Ph.D. Dissertation

**Norwegian University of Science and Technology - NTNU
Department of Production and Quality Engineering**

Trondheim, Norway

Preface

This dissertation is written in partial fulfillment of requirements for the degree of DOKTOR INGENIØR (Dr. Ing.) at the Norwegian University of Science and Technology (NTNU), Department of Production and Quality Engineering (IPK). The bulk of the research presented was conducted at the Intelligent Coordination and Logistics Laboratory (ICLL) of the Robotics Institute at Carnegie Mellon University (CMU).

Professor Øyvind Bjørke and Professor Kesheng Wang at IPK have been my supervisors. My host and advisor during my stay at CMU has been Doctor Norman M. Sadeh, Co-Director of ICLL. The first three years of research were supported financially by a grant from the Research Council of Norway and ICLL has financed most of my stay at CMU.

It gives me great pleasure to take this opportunity to thank some of the people without whom the completion of this dissertation would not have been possible. First of all, I would like to express my deepest gratitude to Norman for the support and guidance he has provided and for challenging me to pursue supply chain coordination as a research topic, a decision that I have never regretted. Since Norman has always taken pride in the quality of the research in his group, he expected a high standard of excellence from my research as well. He was always readily available to answer questions and provided excellent help, even when he had to spend late night hours in doing so. Norman's involvement and crucial suggestions have had a decisive influence on my work and contributed greatly to improving the quality of this dissertation.

I am no less grateful to thank Øyvind and Kesheng for accepting me as their student, for the confidence in me they demonstrated by permitting me to pursue my research orientation, and for patiently accepting the many progress delays. In addition, I must thank Professor Asbjørn Rolstadås at IPK for providing the necessary contacts that made my stay at CMU possible. My thanks also go to Stephen F. Smith, Director of ICLL, for providing office space and the financial support that kept me going during the last months of my stay at CMU.

David Hildum and Allen Tseng have had a great intellectual influence on my research through the many hours of discussion I have enjoyed with them. Their companionship has meant much to me. Among the others who have influenced the course of this work, I would like to mention in particular Kyoung Jun Lee for our early discussions on supply chain coordination issues, Ora Lassila for our discussions on object-oriented design and the implementation of generic scheduling systems, Geir Hasle and Johan Haavardtun for their valuable comments and suggestions

concerning a developed draft of the text, and Andrew Blasko for his help in improving the readability of the final version of this dissertation. Warm thanks go out to all of you and to the rest of my friends and colleagues at NTNU, CMU and SINTEF. It has been a pleasure knowing and working with you all.

My love goes to Elisabeth, whose patience with my physical and mental absence from home has put me in an unlimited debt of gratitude to her, and to Lindy and Emil, who took their father's preoccupation so easily in their stride.

Oslo, November 1, 1998

Dag Kjenstad

Abstract

As we move towards an increasingly global market economy, companies are forced to focus on the production of high-value-adding components. Increase in customer expectations in terms of cost and services has put industry under pressure to become more agile and provide timely yet cost effective deliveries under highly dynamic market and supply conditions. Just-in-time production methods have become popular for reaching these goals, but this also exposes manufacturers to the reliability of their suppliers and, in turn, increases the interdependency between manufacturers and their suppliers.

This thesis is concerned with coordination aspects of supply chain management and, in particular, explores lateral coordination across the supply chain. We propose a new framework for supply chain coordination, including mechanisms and policies that leverage finite capacity scheduling to provide more effective coordination. These mechanisms are studied under a number of different supply chain configurations, supplier-customer relationships, load conditions, and degrees of uncertainty.

The proposed coordination framework is designed for decentralized systems of self-interested and rational entities. It is based on agent technology, where several software agents, each responsible for a particular supply chain entity (e.g., a shop or an entire plant), cooperate and coordinate to maintain consistent schedules. Key advantages of this approach are its assumption of decentralized control (i.e., each supply chain entity, modeled as a software agent, is an autonomous entity), the possibility of concurrent or asynchronous execution, the flexibility of control mechanisms, and the ability to reconfigure and extend the supply chain model.

We have tested and compared the performance of our coordination policies according to a number of indicators, such as profit (sales revenue minus costs), leadtimes, customer satisfaction, and the ability to accurately forecast order completion. Our empirical experiments indicate, with high levels of statistical significance, that policies which synchronize finite capacity schedules across the supply chain can reduce the number of tardy orders by up to 50 percent, cut leadtimes by up to 30 percent, and provide a significant increase in profit over traditional leadtime-based coordination approaches.

Table of Contents

| | |
|--|-----------|
| 1. INTRODUCTION | 1 |
| 1.1 Informal problem description | 1 |
| 1.2 Summary of contributions | 4 |
| 1.3 Overview of the thesis | 6 |
| 2. FINITE CAPACITY SCHEDULING | 7 |
| 2.1 Scheduling techniques | 8 |
| 2.2 Finite capacity scheduling and the real world | 17 |
| 2.3 The Micro-Boss scheduling system | 19 |
| 2.3.1 The micro-opportunistic search procedure | 19 |
| 2.3.2 The Micro-Boss modeling framework | 23 |
| 3. SUPPLY CHAIN MANAGEMENT | 31 |
| 3.1 Introduction to supply chain management | 31 |
| 3.2 Distributed AI for supply chain management | 38 |
| 3.3 Supply chain modeling | 41 |
| 3.3.1 Supply chain modeling frameworks | 42 |
| 3.3.2 Supply chain architectures | 43 |
| 3.4 Aspects of autonomy | 46 |
| 3.4.1 Stability | 46 |
| 3.4.2 Optimality | 47 |
| 3.4.3 Fairness | 47 |
| 3.4.4 Negotiation | 48 |
| 3.4.5 Veracity | 49 |
| 4. A FRAMEWORK FOR SUPPLY CHAIN COORDINATION | 53 |
| 4.1 The conceptual model | 53 |
| 4.2 Context mechanism | 54 |
| 4.3 Integration of bill-of-material and process plan | 55 |
| 4.4 Connectors | 56 |

| | | |
|-----------|--|-----------|
| 4.5 | Interaction with external suppliers and customers | 58 |
| 4.6 | Coordination | 59 |
| 4.6.1 | Asynchronous coordination by means of “unresolved issues” | 59 |
| 4.6.2 | Coordination by means of high-level agents | 60 |
| 5. | SUPPLY CHAIN COORDINATION POLICIES | 63 |
| 5.1 | Just-in-time policies | 63 |
| 5.1.1 | The just-in-time leadtime-based policy (JIT-Lead) | 63 |
| 5.1.2 | The just-in-time synchronization policy (JIT-Sync) | 66 |
| 5.2 | Safety leadtime policies | 66 |
| 5.2.1 | The time-buffered leadtime-based policy (Buf-Lead) | 67 |
| 5.2.2 | The time-buffered synchronization policy (Buf-Sync) | 69 |
| 5.3 | Bid refusal policies | 70 |
| 5.3.1 | The leadtime-based bid refusal policy (Lead-Ref) | 70 |
| 5.3.2 | The synchronization-based bid refusal policy (Sync-Ref) | 71 |
| 5.3.3 | The finite capacity-based bid refusal policy (FCS-Ref) | 71 |
| 5.4 | Promise date negotiation policies | 71 |
| 5.4.1 | The leadtime-based promise date negotiation policy (Lead-Neg) | 72 |
| 5.4.2 | The finite capacity-based promise date negotiation policy (FCS-Neg) | 72 |
| 5.4.3 | The synchronization-based promise date negotiation policy (Sync-Neg) | 72 |
| 5.4.4 | Promise date negotiation with competition | 72 |
| 6. | EMPIRICAL EVALUATION | 75 |
| 6.1 | The scope of the experiments | 75 |
| 6.2 | Evaluation of supply chain performance | 77 |
| 6.2.1 | Statistical confidence | 81 |
| 6.3 | The simulation testbed | 82 |
| 6.3.1 | Sources of randomness | 83 |
| 6.3.2 | Estimation of the nominal load | 88 |
| 6.3.3 | The schedule synchronization controller | 88 |
| 6.3.4 | Initialization | 89 |
| 6.3.5 | The simulation cycle | 89 |
| 6.4 | Results for a simple two-tier model | 90 |
| 6.4.1 | Just-in-time policies | 91 |
| 6.4.2 | Safety leadtime | 93 |

| | | |
|-----------|---|------------|
| 6.4.3 | Refusal of bid requests | 94 |
| 6.4.4 | Promise date negotiation | 97 |
| 6.4.5 | Promise date negotiation under competition | 99 |
| 6.5 | Results for alternative supply chain configurations | 101 |
| 6.5.1 | Long supply chains | 101 |
| 6.5.2 | Multiple customers | 103 |
| 6.5.3 | Multiple suppliers | 105 |
| 6.5.4 | A larger example | 106 |
| 6.6 | Results for variations of external conditions | 108 |
| 6.6.1 | Variations in nominal load | 108 |
| 6.6.2 | Variations in degree of uncertainty | 110 |
| 7. | CONCLUSIONS | 115 |
| 7.1 | Thesis summary and conclusions | 115 |
| 7.2 | Directions for future research | 117 |
| | REFERENCES | 119 |
| | APPENDIX A: PARAMETER SETTINGS FOR THE EXPERIMENTS | 135 |
| | APPENDIX B: AUXILIARY EXPERIMENTS | 143 |
| | APPENDIX C: LIST OF SYMBOLS | 149 |

1. Introduction

1.1 Informal problem description

A supply chain can be defined as a world-wide network of business entities, such as suppliers, factories, warehouses, distribution centers, and retailers, through which raw materials are acquired from suppliers, transformed, and delivered to customers. Information and material flow between these autonomous or semi-autonomous sites, all of which strive for profit maximization.

Today's movement towards an increasingly global market economy is constantly forcing companies to focus on the production of high-value-adding core components. The increase in customer expectations in terms of cost, quality, and services has put industry under pressure to become more agile and provide timely yet cost effective deliveries under highly dynamic market and supply conditions. Just-in-time (JIT) production methods have become popular for reaching these goals, but this also exposes manufacturers to the reliability of their suppliers and, in turn, increases the interdependency between manufacturers and their suppliers. A common criticism of JIT philosophy has been that manufacturers use it as a means to transfer their own inefficiencies to their suppliers (Hall 1983 p. 202; Helper 1991; Romero 1991). Suppliers are forced to maintain large and costly buffers of finished goods inventory in warehouses dedicated to the customer, a burden on suppliers that can put them out of business. This view is contradictory to the basic JIT philosophy, which is to expose the system to problems, search for the causes of problems, and correct them

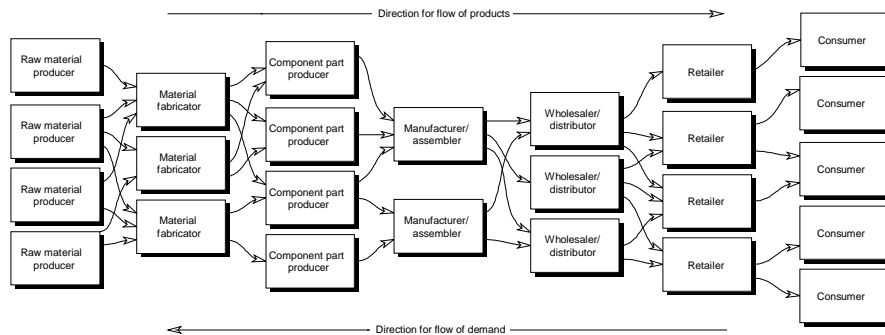


Figure 1: An example of a supply chain network. The network is split into tiers of business entities, and information and material flow from tier to tier in both directions.

at their source. True JIT practices require coordination across the entire supply chain. It is thus often argued that it is necessary to establish partnership agreements which make possible continuous improvement in those aspects critical for getting a competitive edge, thereby enhancing the long-term competitiveness of each party involved. The introduction of finite capacity scheduling has already been proven to be of assistance in optimizing production within single plants, but at the same time it reveals the inefficiencies in relations between different facilities. The competitiveness of a company is therefore increasingly tied to the dynamics of the supply chain. The challenge then becomes how to identify and deal with these inefficiencies.

Supply chain management is a potential source of savings that cannot be ignored. Logistics costs often account for more than 30 percent of total costs. The following specific examples illustrate the magnitude of benefits that become possible as a result of improved supply chain management practice:

- Levi Strauss plans to deliver 95 percent of their orders on time as the result of a major re-engineering of their supply chain based upon quick and accurate response, partnerships with suppliers and retailers, and the redesign of distribution (Knill 1994).
- National Semiconductor has managed to reduce delivery time by 47 percent and distribution cost by 2.5 percent, leading to increased sales of 34 percent, through effective supply chain re-engineering (Henkoff 1994).
- Extensive research sponsored by the Food Marketing Institute has demonstrated that efficient consumer response can save close to \$10 billion in operating costs for grocery supply chains across United States industry (Narayanan 1994). Efficient consumer response encompasses integrated electronic data interchange, continuous replenishment, flow-through distribution, activity-based costing, and computer assisted ordering.
- Digital Equipment Corporation initiated a project in 1989 to create a global model of their supply chain for analytical purposes. These studies led to a major restructuring of Digital's physical supply chain. Arntzen *et al.* (1995) estimates a \$1 billion reduction in cumulative costs, a \$400 million reduction in asset costs, and a 500 percent improvement in unit production (fewer people making more products) due to these restructuring efforts.

Research in the field of supply chain management has provided valuable insight into supply chain issues that has led to a significant change in current industry practices. However, research on operational aspects of decision-making, including supply chain coordination issues, has been of limited use in providing applicable

mechanisms for the effective coordination of finite capacity schedules between autonomous and self-interested supply chain partners.¹ In that many of the proposed solutions force the individual entities of the supply chain into a strict hierarchical model of cooperative decision-making, they are inappropriate for distributed supply chains. They ignore the fact that each organization in an inter-organizational supply chain has its own set of objectives (e.g., meeting expectations from several supply chains at the same time) and should be viewed as self-interested, or even as antagonistic rather than cooperative in respect to other organizations in the chain. We claim that inter-organizational supply chains may have objectives that are likely to conflict with the “anarchic” objectives of individual organizations, and hence that mechanisms for centralized supply chain scheduling (e.g., by means of hierarchical models) only apply for intra-organizational supply chains.

Ongoing developments in computer and information technology are continually revolutionizing the world of business management. *Inter-organizational systems* (IOS) for the integration of companies into *extended enterprises* or *virtual enterprises* have been proposed. *Electronic data interchange* (EDI) has become popular due to its speed, reliability, and cost effectiveness, while *electronic commerce* (EC) standards are evolving for communication between trading partners. The expected benefits of information technology applied to supply chain coordination are tremendous and have, for example, led to major investments in information technology for automatic information retrieval and transfer in firms like Kmart (Mandell 1991a), Caterpillar (Mandell 1991b), and Wal-Mart (Reid 1995). Such systems bypass intervening manual steps by direct computer-to-computer transfer and are expected to be one of the keys to reducing swings in requirements and increasing the speed of delivery. An analysis of shipment data in the American automobile industry (Srinivasan *et al.* 1994) supports these expectations, concluding that sharing of JIT schedules and establishing integrated EDI links are related to a significant reduction in the level of shipment discrepancies, especially when the part variety is high.

Research on intelligent software agents has offered new tools that are well suited for dealing with distributed problem-solving and coordination between autonomous sub-systems. These tools allow sub-systems to be represented as agents that can coordinate their activities through message passing. Key advantages of this approach when applied to a supply chain model are distributed autonomy, the possibility of real-time, parallel and asynchronous execution, flexibility of the control

¹ By *self-interested* we mean an entity who cares more about its own objectives than system-wide objectives and, therefore, would never be willing to accept a lower utility just to increase the system's sum.

mechanisms, and the ability to facilitate reconfigurations of and extensions to the model.

The aim of this thesis is to propose a new framework for operational decision-making within supply chain management. This framework utilizes mechanisms and policies that leverage finite capacity scheduling to provide more effective coordination across the supply chain. It emphasizes the distributed autonomy between individual rational and self-interested supply chain entities. The proposed lateral coordination policies therefore de-emphasize the overall optimization of schedules and instead focus on a coordination of locally optimized schedules through the exchange of temporal constraints.

One of the main objectives of academic research of the type carried out in this dissertation is to propose and study concepts and ideas that may constitute practical operational principles for future generations of systems. Hence, those familiar with today's practices may view our research as being based on certain radical and unconventional assumptions. We by no means claim that the results obtained are applicable to or embrace all manufacturing environments. Still, we hope that the present study will convince readers from industry about the potential benefits of a synchronized supply chain and motivate them to assess the situation within their businesses to see whether this is a fruitful way to approach world class manufacturing.

1.2 Summary of contributions

The main contributions of this dissertation can be summarized as follows:

- We have proposed and evaluated coordination policies applicable to both intra- and inter-organizational supply chains under different conditions. In contrast to most research in the area, these policies do not assume hierarchical authority structures for decision-making and thus appear especially powerful for coordinating schedules among autonomous and self-interested supply chain partners.
- We have obtained experimental results for a make-to-order environment that demonstrate the benefits of a synchronized supply chain over traditional customer-supplier relations. Our results indicate that a synchronized supply chain is superior in forecasting the completion date for bids and in-process orders. These forecasts are used for decision support during bid negotiation, such as by only allowing those bids that are expected to be profitable to be submitted, or by proposing alternate and more realistic delivery dates for the bids. Synchronized coordination

through just-in-time exchange of detailed schedule information also helps the supply chain participants prioritize more effectively. Altogether, this results in improved due-date performance, which again can be reflected in the overall profit to share between the supply chain participants.

- The experimental results also illustrate the benefit of using a finite capacity scheduling system as a decision support tool during the bid-preparation process. Finite capacity scheduling makes it possible to obtain qualified leadtime estimates based on the current load and capacity constraints of a factory. These estimates can then be used to derive prospective and realistic delivery dates for potential orders.
- While the traditional perspective of supply chain management views customer-supplier relationships in terms of both parties competing for profit margins, the coordinated supply chain is based on a relationship in terms of partnership-in-profit creation. The research in this dissertation provides additional insight into aspects of operational supply chain management that highlights the benefits of the partnership-in-profit perspective. An antagonistic and competitive attitude towards supply chain partners will not only be destructive for the overall supply chain, but may also be disadvantageous for the entity itself.
- Our model considers the execution of optimized finite capacity schedules in the analysis of supply chains. If analytical work in the supply chain literature considers finite capacity at all, it relies on local priority rules (queuing type models) at individual work centers. It is therefore of restricted interest for supply chains where production entities run their operations upon the basis of more sophisticated scheduling techniques.
- The dominant view of supply chain coordinated scheduling tends to assume that scheduling can be performed as an off-line process where the goal is to achieve schedules that are globally consistent. We claim that this approach will not work in practical settings, and introduce the alternative idea of schedule revision as a continuous and dynamic process across the supply chain. Local schedules are periodically and asynchronously updated to take the most updated information received from other supply chain entities into account. A fresh schedule does not necessarily need to be consistent in respect to all information received from outside supply chain partners. However, recurrent schedule updates will tend to minimize the severity of inconsistencies.

1.3 Overview of the thesis

The remainder of this dissertation is divided into six chapters. Chapter 2 begins with an overview of different approaches to finite capacity scheduling and a discussion of challenges that need to be faced in order to enable effective use of finite capacity scheduling systems in practical settings. Thereafter, the chapter presents the Micro-Boss system for factory scheduling — the system used within our experiments. Chapter 3 introduces supply chain management in general and emphasizes aspects of supply chain management that relate to our experiments. Chapter 4 presents a decision support framework for supply chain management with an emphasis on mechanisms for operational lateral coordination. Chapter 5 proposes a number of supply chain coordination policies, each addressing when, what, and with whom to communicate between supply chain participants under different assumptions. The coordination policies are evaluated empirically in Chapter 6. This chapter first introduces the supply chain coordination experiments with a description of the assumptions made, how the experiments have been carried out in our testbed, and how the experimental results have been collected and evaluated. It then presents the experimental results obtained. The discussion begins with an evaluation of the experiments on the basis of a simple model of the supply chain and its environment. These experiments enable us to isolate and highlight aspects of conceptual interest. The model is then gradually enhanced towards more realistic assumptions and supply chain configurations. Chapter 7 concludes this dissertation with a final set of remarks.

2. Finite capacity scheduling

Within the field of production management, production planning and control comprise the activities necessary to carry out production. Solberg (1989) identifies three paradigms for production planning and control:

- The *optimization paradigm* is based on the realization that the overall performance of a manufacturing system is the result of individual decisions, and that decision-makers are *computationally limited*. Optimized production schedules are therefore created up front as a guideline for the decision-making process.
- The *data processing paradigm* focuses on data management issues and the development of computer-based methods to organize and manipulate vast amounts of data. The multitude of today's MRP systems (Fox 1984) originated from this paradigm.
- The *control paradigm* focuses on control aspects (by means of cybernetics theory) to obtain a stable production system in the presence of disturbances, such as the use of Kanban cards (Ohno 1988; Japan Management Association 1986).

The objective of this chapter is to introduce the reader to production scheduling within the optimization paradigm, often referred to as *finite capacity scheduling*. Finite capacity scheduling is a means of choosing a course of actions before performing it. It provides guidance in achieving global coherence in the process of local decision-making, such as how to use a limited set of shared resources to effectively meet global objectives or how to effectively react when unexpected events force changes. In practice, scheduling problems arising in manufacturing systems are discrete, distributed, dynamic, and stochastic, and turn out to be of a combinatorial nature. Simple algorithms for finite capacity scheduling are therefore not available with the exception of certain strictly defined and simplified situations (French 1982; Baker 1992).

The chapter is organized into three parts. The first part will present a general survey of state of the art approaches to finite capacity scheduling. Given the primary objective of the chapter and the multitude of approaches that have been reported, the discussion of each individual approach is kept to a minimum. The second part comprises a discussion of challenges that must be met before finite capacity scheduling can be effectively applied in real world manufacturing environments. The

third part is dedicated to the Micro-Boss scheduling system, which is a core element of the supply chain coordination experiments presented in later chapters.

2.1 Scheduling techniques

Over the past few decades, a large number of efforts have sought to investigate finite capacity scheduling. Such surveys of state of the art techniques within finite capacity scheduling can be found in Smith 1992; Blazewicz *et al.* 1993; Suresh and Chaudhuri 1993; Dorn and Froeschl 1993; Szelke and Kerr 1994; Zweben and Fox 1994. We will now briefly and systematically review these different approaches, beginning with classical operations research (OR) approaches and afterwards reviewing various artificial intelligence (AI) approaches and systems.

Early OR work focused on finding optimal solutions according to single objective functions (e.g., minimization of makespan). One approach was to categorize scheduling problems into strictly defined classes of problems (e.g., single-machine sequencing or two-machine flow-shop to minimize total flow-time) where simple algorithms are applicable (Johnson 1954; Baker 1992). A more general approach was to use a *mixed integer linear programming* formulation of the problem (Pritsker *et al.* 1969). The formulation comprises a collection of variables, constraints on their possible values, and an objective function to be either minimized or maximized in the process of assigning values to the variables. The objective function may encode a single scheduling objective, or it may attempt to satisfy a collection of multiple objectives (e.g., minimization of both order tardiness and amount of changeover activity required). Among the optimizing solution methods we also find *branch-and-bound*, *enumeration*, and *dynamic programming*. A general survey of branch-and-bound techniques is provided in Lawler and Wood (1966), a comparison of different enumeration approaches can be found in Patterson (1984), and an early dynamic programming approach for sequencing problems is described in Held and Karp (1962). These methods are also well covered in French (1982) and Baker (1992).

The high complexity of a large portion of real world scheduling problems makes it practically impossible to find optimal solutions. We maintain that the optimal solution is *computationally intractable*. This realization led to a shift of effort within OR towards the development of methods able to produce *near-optimal* solutions with significantly less computational expense than the optimizing methods. The most common method in this regard involves the use of *heuristic scheduling rules*. These rules specify what decision to make in a particular situation by rating alternatives based on local and myopic considerations, for example, by scheduling

activities with the least available amount of slack before activities with more slack. A survey of heuristic scheduling rules is presented in Panwalkar and Iskander (1977). An attempt to identify appropriate scheduling heuristics for different classes of scheduling problems is given in Kurtulus and Davis (1982). Boctor (1990) groups multiple scheduling heuristics together in an attempt to increase the chance of producing near-optimal schedules. More recent developments in scheduling heuristics are presented in Morton and Pentico (1993).

A different OR approach for the creation of near-optimal solutions is presented by Della Croce *et al.* (1993). They consider a cellular manufacturing system where the overall scheduling problem is decomposed into *modules* (e.g., corresponding to a shop, cell or machine depending on the level of resolution considered) and apply a *Lagrangian relaxation technique* to coordinate the schedules between the modules. Local due dates and release times within a module are defined by start and completion times of lots in neighboring modules in the material flow. The optimization of each sub problem is performed independently, and temporal constraint violations are communicated to a higher layer control module, which in turn returns Lagrangian multipliers obtained by solving the dual problem recursively. To prevent oscillation in the solution, quadratic penalty costs associated with violation of temporal constraints are introduced into the Lagrangian function. This approach has certain advantages for supply chain coordination in its ability to support distributed systems. These aspects will be further addressed in Chapter 3.

The approach of OPT (Optimized Production Technology) (Jacobs 1984; Fox 1987) takes a *bottleneck-centered* approach to scheduling. The OPT philosophy emphasizes the need to distinguish between bottleneck and non-bottleneck resources. A module of the OPT system called SERVE produces an initial infinite capacity schedule by working backwards from the job due dates. This schedule helps detect the bottleneck resources. The OPT module will then generate a forward finite capacity schedule that optimizes the utilization of these bottlenecks. The resulting bottleneck schedule is passed back to the SERVE module, which schedules the non-bottleneck activities while trying to minimize inventory. The OPT system represents one of the first successful scheduling approaches. However, the most common criticisms of OPT lie in its reliance on static bottlenecks, the proprietary (unpublished) nature of the scheduling algorithm, and the lack of support for interactive schedule editing.

AI approaches to scheduling can be distinguished from OR approaches by their focus on representing the diversity of characteristics and constraints found in practical problems. In certain cases, these efforts have provided frameworks for making traditional OR-based techniques usable in practical settings. In other cases,

novel techniques have been provided that offer new opportunities within finite capacity scheduling. These efforts can be characterized along the following dimensions:

- Decision-making level — whether the system is aimed at strategic, tactical or operational decision-making, or at the integration of decision-making levels into a hierarchical model.
- Application domain — the manufacturing environment at which the system is aimed, such as job-shop, flow-shop, flexible manufacturing systems, or distributed manufacturing systems.
- Real time support — the degree to which the system is able to represent and respond to real-time changes. Some systems take a purely *predictive* scheduling orientation, where the focus is on producing a schedule that solves a static problem description. Other systems take uncertainty into account *proactively* and focus on producing robust schedules, that is, schedules that are resilient enough to absorb sources of uncertainty without being invalidated. Systems with a *reactive* scheduling orientation are able to dynamically revise the predictive schedule in reaction to changes. The degree of real time support is thus related to the ability to capture and represent feedback from the shop-floor and to the speed with which a revised solution can be proposed as changes are introduced.
- Scheduling strategy — that is, job-based, resource-based, activity-based, or opportunistic decomposition of the search space.²
- AI approach — that is, interactive or mixed initiative scheduling, rule-based scheduling, simulation-based scheduling, constraint-based scheduling, fuzzy scheduling, iterative scheduling, case-based scheduling, planning and scheduling, distributed scheduling, or hybrids of these approaches.²

Early interactive systems were simply designed as a visualization tool to help the user in manually creating and modifying schedules. The transformation of human knowledge into a computer-understandable format was, in the early years of knowledge-based scheduling, identified as a bottleneck in the design of systems (Feigenbaum 1977). However, interactive systems allow the human scheduler to be an integral part of the scheduling process and they early on found their way into real applications. As the size of the scheduling problem grows, the burden on the user to create and verify the schedule increases significantly, and may easily overwhelm him/her (Fox and Smith 1984). Interactive systems have later been enhanced to work in conjunction with computerized methods to build initial schedules and to measure

² These expressions will be defined later in this section.

the effect of manual changes with respect to schedule validity and quality (Elleby *et al.* 1989; Meng and Sullivan 1991). At the user interface level, research has advocated the concept of “electronic leitstand” (Adelsberger and Kanet 1991; Haavardtun and Kjenstad 1995; Haavardtun 1995), a production control “command center” that combines graphical displays with constraint checking and scheduling capabilities to provide a variety of editable views of the current production schedule and shop-floor status.

Rule-based approaches to scheduling have been pursued in so-called expert systems (Bensana *et al.* 1986; Kerr and Ebsary 1988). The objective under this approach has been to mimic the decision-making of a human expert. The knowledge is collected and stored as “if-then” rules in a *knowledge base*. An *inference engine* controls the process of incrementally selecting and applying rules from the knowledge base, normally in a forward-chaining (constructive) manner towards a satisfactory result. Expert systems have been successful in solving problems where the number of possibilities at each step is fairly small. This is not the case in finite capacity scheduling, where rule-based approaches can easily result in sub optimal solutions.³

Another knowledge-based approach to scheduling has been explored by discrete event simulation (Jain *et al.* 1989; Wyman 1991; Drake and Smith 1996; Kunnathur *et al.* 1996). In this approach, a schedule is created by simulating the execution of an appropriate dispatch heuristic, or, in conjunction with a rule-based approach, for selecting appropriate activities for execution. A component that makes this approach differ from the heuristic method of OR is the use of powerful modeling and knowledge-representation techniques to allow complex scheduling constraints and opportunities to be modeled, such as alternative process plans and product mix constraints. Another advantage of simulation lies in the ability to account for real world events of a probabilistic nature and thus produce robust proactive schedules. On the other hand, a disadvantage lies in its inability to make an infeasible schedule feasible without re-generating a completely new schedule. Simulation-based scheduling is therefore of limited usefulness in dynamic environments where schedule continuity is desired.

A good schedule is typically a schedule that is able to meet or balance a range of conflicting objectives and preferences. The perspective of scheduling as a constraint-driven process has therefore lately become more and more dominant. Constraint-based approaches include a constraint management component in addition to the search component. The constraint manager is responsible for the deduction of a

³ This argument was presented in the discussion regarding interactive systems. A system that mimics a human scheduler does not necessarily produce good schedules.

tight upper bound description of feasible values for each of the variables in the problem. Instead of relying purely on a generate-and-test approach, a constraint-based approach may allow for a guided least commitment style of scheduling, that is, the generation of solutions that retain degrees of freedom where constraints allow. It also makes possible the early detection of inconsistencies. Special conflict resolution techniques, for example, chronological backtracking (Sadeh *et al.* 1995), dynamic backtracking (Ginsberg 1993), and reactive local repair heuristics (Ow *et al.* 1988a), have been developed to handle such situations.

The ISIS system (Fox 1983; Fox and Smith 1984) represents the first generic constraint-based system for scheduling. It emphasizes a broad view of constraints that recognizes the conflicting and negotiable nature of many requirements and objectives, and focuses on the representation and use of knowledge about constraints to support effective conflict resolution and relaxation. This system implements a four-phase hierarchical scheduling approach: (1) select the job with the highest priority from the set of jobs; (2) generate time bounds for each of the alternative resources on which the activities of the job may be performed; (3) search for the most promising plan among the set of candidate process plans; and (4) determine reservation times for each activity of the job. The four phases are repeated until all jobs are scheduled. The third phase applies a beam search to rank and prune the set of candidate plans. It sequentially allocates resources to activities by either extending forward from the first activity or backward from the last. When conflicts are detected, the relevant constraints are identified and relaxed, and the process plan involved in the conflict is corrected. Unfortunately, ISIS's reliance on a single job-centered scheduling perspective has a disadvantage in that decisions are made that cannot anticipate later unscheduled jobs. The ability to deal with inter-job concerns such as resource conflicts or sequence-dependent setups is therefore limited. The major contribution of this work is in the area of constraint formulation and the use of constraints to narrow the search space. Despite the above limitations, systems based on job-centered techniques, for example, JOBCODE, have made their way into operational use (Hastings and Yeh 1990).

The OPIS system (Smith 1989) represents a major step forward in the development of knowledge-based scheduling systems by identifying and demonstrating the utility of viewing scheduling from *multiple perspectives*. The consultation of a resource-based perspective in addition to the job-based perspective of ISIS has proven useful in providing an informed view of the overall problem. The opportunistic nature of the scheduling approach used by OPIS derives from its ability to shift the focus of attention between the various scheduling perspectives. At the start of each scheduling cycle, OPIS performs a capacity analysis based on the

updated demand for resources. If there is a sufficient level of contention for any particular resource, a resource-based scheduling strategy is invoked to satisfy all outstanding requests for that resource. Otherwise, the job-based strategy of ISIS is invoked. The job-based scheduler completely finishes the schedule for a job, either forward or backward from previous resource-based scheduling decisions. The resource-based scheduler completely finishes the schedule for a resource class. OPIS introduced several novel techniques for constraint-based scheduling. The first, opportunistic problem decomposition into multiple perspectives, has already been described. OPIS also introduced techniques for real-time support, such as reactive scheduling through schedule repair. Finally, OPIS was designed using a blackboard architecture (Erman *et al.* 1980) that emphasizes the modular integration of analysis and scheduling methods and a rich underlying structure for modeling manufacturing environments.

While systems like ISIS and OPIS include full implementations of the constraint manager, other systems rely on programming languages or libraries based on generalized constraint propagation techniques (Colmerauer 1990; Dincbas *et al.* 1988; Jaffar *et al.* 1990; Le Pape 1994). The advantage of this approach is that the system designer can focus on modeling the appropriate constraints of the problem and simply select the appropriate strategy to solve the problem. One disadvantage of this approach is that generalized techniques often sacrifice performance since they make it difficult to use domain specific knowledge in their implementation. Another disadvantage is that the system designer has limited control over the capabilities of the constraint manager and is restricted to what the generalized techniques can offer. Work on constraint propagation techniques for scheduling can be found in Rit (1986), Dechter *et al.* (1991), Le Pape (1991), Le Provost and Wallace (1992), and Nuijten (1994). Basic constraint satisfaction search techniques are limiting in one important aspect of scheduling, namely, problems are more often than not over-constrained (e.g., all due-dates cannot be met). The problem then becomes how to relax constraints in order to obtain a good compromise. One general approach to this is through the work of constraint hierarchies (Borning *et al.* 1992), where the importance of various constraints can be expressed and constraints may selectively be relaxed according to predefined priorities. However, the author is not familiar with any scheduling system based on Borning's constraint hierarchy approach.

It can be argued against most predictive scheduling techniques that they take the data in the model for granted and have no mechanism to represent the uncertainty or fuzziness that the data really includes. Significant computational effort may be wasted in creating crisp assignments of activities that are likely to be invalidated by shop-floor contingencies. For example, the farther into the future we look, the less

effort we should spend on dealing with minor details. This perspective has led to approaches in scheduling that explicitly factor uncertainty into the generated schedule. By using a fuzzy-set approach to scheduling, temporal constraints, activity duration, resource capacity, etc., may be represented as sets of fuzzy numbers (ranges) with associated levels of confidence as to membership in the set. Kerr and Walker (1989) present the FSS system, where fuzzy constraint management techniques are combined with a rule-based approach and a resource perspective to scheduling in which fuzzy constraints must have a minimum threshold degree of satisfaction. Bensana *et al.* (1988) utilize fuzzy sets in the OPAL system as the basis for a weighted voting scheme in which advice for alternative decisions is accumulated from different rules. Other approaches are reported in Dorn *et al.* (1994), Muscettola and Smith (1987), and Chiang and Fox (1990). One advantage of fuzzy scheduling lies in its ability to focus and limit the computational effort to significant scheduling decisions. However, a disadvantage lies in the increased computational expense of fuzzy constraint propagation compared with its deterministic counterpart.

Iterative and repair-based scheduling techniques represent a wide range of techniques that, instead of navigating in the search space of partial schedules, navigate across a set of complete (but possibly inconsistent) schedules using a neighborhood search technique. The search may terminate either when an acceptable schedule is found, when a predefined amount of search is completed, or by user interruption. Iterative techniques have an advantage over constructive techniques in that they can easily be implemented as single, general purpose techniques applicable to both predictive- and reactive scheduling. Many of these techniques produce highly optimal solutions if enough time is provided. Among the techniques that have proven efficient for scheduling are simulated annealing (van Laarhoven *et al.* 1992; Nakakuki and Sadeh 1994), genetic algorithms (Mattfeld 1995; Soares 1994; Syswerda and Cerys 1990) and taboo search (Glover and Laguna 1993; Taillard 1994; Nowicki and Smutnicki 1996). A comparative study of iterative techniques is reported by Dorn (1995). Most iterative techniques rely on extensive navigation across the search space. However, taboo search appears particularly appealing since it can be implemented without a random component. The search is then able to escape from local optima by memorizing visited states. The challenges for large-scale problems lie in providing fast and memory-efficient ways to remember a sufficient number of visited states to avoid looping back into already visited, sub-optimal areas of the search space and in identifying effective *neighbor operators* to move from one state to the next.

Ow *et al.* (1988b) take a more deliberate approach, relying less on extensive navigation/search and more on conflict analysis of current solution constraints and coarser search-based repair actions. Another approach to iterative scheduling is taken in CABINS (Miyashita 1994; Miyashita and Sycara 1994). CABINS utilizes case-based reasoning (Kolodner *et al.* 1985) for the acquisition and reuse of scheduling preferences and selection of iterative repair actions. A *case base* containing examples that collectively capture performance tradeoffs under diverse problem solving circumstances is used (1) to map the current problem solving state to an appropriate action in the search procedure, (2) evaluate intermediate repair results, and (3) recover from revision failures. Various cases are recorded into the case base either in a user-directed mode, where the user selects a repair tactic and evaluates its result, or in an automated mode, where CABINS suggests and evaluates a repair tactic but allows the user to override the suggestions. CABINS works within the space of complete and consistent schedules (anytime-executable schedules). Constraint propagation techniques are utilized to propagate the ripple effects of a repair action to the rest of the schedule, which are then reflected in the evaluation metrics of the schedule. Advantages of this approach resides in its ability to achieve situation-sensitive efficient search and improve schedules on the basis of tradeoffs between schedule objectives that are impossible to consolidate realistically in the form of simple objective functions. This approach also represents a natural decision support enhancement for interactive systems. More recently, Miyashita has developed CAMPS, a distributed version of the CABINS system (Miyashita 1998).

Most scheduling approaches rely on totally or partially pre-determined process plans. Some production environments provide the flexibility to manufacture a part in several different ways with respect to resource sequencing and assignments. The inherent constraints of a single pre-compiled process plan may then prohibit the system from producing good schedules. The planning and scheduling activity should therefore be integrated in these environments. Research on integrating AI generative planning techniques with scheduling is reported in Shaw (1988), Wilkins (1989), Muscettola and Smith (1990), and Laborie and Ghallab (1995). Work on integrating resource allocation concerns into the plan selection process is described by Shaw (1988) and by Sadeh *et al.* (1998) in respect to the IP3S (Integrated Process Planning and Production Scheduling) system. IP3S advocates planning and scheduling as a mixed initiative process in which the user can interactively select courses of actions and analyze tradeoffs between alternatives. IP3S is a *blackboard-based* system (Erman *et al.* 1980), where the planning system and the scheduling system operate as separate *knowledge sources*. The state of the solution is at any time summarized in the form of a set of *unresolved issues*. An unresolved issue is an indication that a

particular aspect of the solution is incomplete, inconsistent, or unsatisfactory. Problem solving in IP3S progresses through cycles during which one or more unresolved issues are selected for resolution, a corresponding method of resolution chosen, and the method executed by invoking the appropriate knowledge source. Separate knowledge sources for analysis and diagnosis help with such considerations as the identification of sources of inefficiency in the current solution, and they determine how the solution can most efficiently be improved, for example, whether to generate an alternative process plan for a given part or reschedule activities on a critical resource.

The emergence of Distributed AI (DAI) introduced a new approach to scheduling whereby the problem to be solved is distributed between a set of agents that interact in the decision-making effort. This original stream of DAI research, in which agents collectively cooperate in the process of achieving the system's overall objectives, is known as cooperative problem solving (CPS). System architectures based on CPS allow the total system to be decomposed into agents by structural elements (e.g., corresponding to organizational units) instead of the functional decomposition (e.g., perception, learning, planning, inference engine) that is dominant in traditional AI. This decomposition facilitates parallel and asynchronous execution and interaction of the goal-directed activity.

The BOSS system for distributed scheduling (Smith and Hynynen 1987; Hynynen 1988) utilizes a hierarchical coordination framework to decentralize the OPIS scheduling approach. Vertical coordination allows constraints to be passed between different layers of abstraction of time and capacity constraints. Lateral coordination enables cooperative management of detailed schedules. A similar approach is taken in the DAS scheduling system (Burke and Prosser 1994). The scheduling problem is distributed among agents in a three-level hierarchical framework corresponding to strategic, tactical, and operational decision-making. The agent at the strategic level is responsible for deciding *what* to produce. Work is delegated to agents at the tactical level that select *where* (what resource) to produce the order. Agents at the operational level select *when* to schedule the activities on the given resources. Communication is limited to the passing of messages up and down in the hierarchy, and backtrack search is applied up and down the hierarchy whenever dead ends are reached. An advantage of both BOSS and DAS is the ability to distribute decision-making according to the organizational structure of companies.

A different CPS approach is taken in CORA/COFCAST (Liu 1996). Liu's approach builds on ideas from the CORTES project (Sycara *et al.* 1991; Muscettola *et al.* 1994) in order to distribute the overall problem among simple, reactive, and homogenous agents, that is, each resource and each order is assigned to an agent.

CORA utilizes a coordination mechanism called Constraint Partition & Coordinated Reaction (CP&CR) to solve constraint satisfaction problems, where local schedule repair is applied in an iterative process until a feasible solution is found. For constraint optimization problems (e.g., relaxable due-dates), a mechanism called Anchor & Ascend is applied for the iterative improvement of a schedule. It first focuses on local optimization for a pre-selected bottleneck resource (the anchor agent), and expands this to construct a global feasible solution using CP&CR. COFCAST emphasizes computational efficiency and utilizes dispatch scheduling as the skeleton procedure of the coordination scheme. This procedure is enhanced with a *coordinated forecast* mechanism to improve the dispatch decisions and reducing the myopia of standard dispatch scheduling.

In this section we have surveyed the state of the art techniques for scheduling that have emerged from the OR and AI communities. Recently, a new focus of scheduling has emerged in which the view of scheduling is broadened to comprise complete *supply chains* or *virtual enterprises*. One challenge for these approaches is how to achieve the effective coordination of autonomous, and often self-interested, entities within the overall system. These challenges will be further discussed later in this dissertation.

2.2 Finite capacity scheduling and the real world

For many years, the number of finite capacity scheduling approaches that found their way into practical industrial applications was restricted due to a gap between what research could offer and what industry needed. This incompatibility was due to limited abilities both within the scheduling systems and in the infrastructure of companies to adapt to the new technologies. Only companies that could afford spending large amounts of money on developing systems tailored for internal use managed to successfully implement finite capacity scheduling in their environment. However, the market for general scheduling applications now seems to be growing at an ever faster rate. Rapid technological advances in computing offer opportunities to present factory models for finite capacity scheduling in more detail than in the past. Early scheduling systems were forced to model resources, activities, and products in a very simplified manner due to limitations in available memory and computational speed. These computational limitations are now virtually gone, and today's scheduling systems can benefit from computers with soaring performance.

However, a long maturity process must take place before a company is ready to benefit from the use of finite capacity scheduling. The creation of realistic schedules requires access to correct and updated data concerning the activities to be

scheduled and the constraints that the schedule must obey. A prerequisite for the effective use of finite capacity scheduling is thus the presence of an *inventory control system* and a *shop-floor tracking system* in that these systems ensure quick access to updated information.⁴ An indirect benefit of having these systems in house and operational before the introduction of finite capacity scheduling is also the possibility of establishing an accurate performance baseline with which to compare performance improvement.

The next challenge is to create a model that captures essential characteristics of the manufacturing environment in a format that the scheduling system can understand. The essential characteristics for a scheduling system are those attributes that limit the performance of the manufacturing system. These may include:

- Production management constraints that limit the time windows within which certain activities can take place, such as the availability of raw material and other resources or technological limitations in the sequence within which activities can take place.
- Resource characteristics, including capacity limitations for essential resources (such as machines, tools, labor) by means of resource calendars and mode of operation (unary capacity, batch capacity, multiple capacity, setup requirements, etc.).

It is also essential to investigate opportunities that the manufacturing environment possesses, such as:

- Freedom to execute activities in alternative sequences.
- Opportunity to execute activities using alternative resources, such as through the organization of resources into pools of identical or similar sub resources.
- Opportunity to overlap activities due to reduced transfer batch sizes between workcenters.
- Opportunity to assign resources to only parts of an activity's duration (e.g., labor requirements for running a semi-automatic machine).
- Freedom to negotiate constraints whenever necessary, such as by policies for the use of overtime and opportunities for negotiating delivery dates with suppliers and customers.

⁴ Shop-floor tracking systems are today often referred to as MES (Manufacturing Execution Systems). MES are information systems that reside on the plant floor, between the planning system in offices and the direct industrial controls at the process itself.

Once the specification of the model is completed, a scheduling system may be selected that is capable of representing the essential characteristics and able to adapt or reconfigure itself to changes in the manufacturing system environment as they are introduced. Last, but not least, the selected system should be able to produce highly optimal schedules for the given environment within a reasonable amount of time.

2.3 The Micro-Boss scheduling system

Micro-Boss is a finite capacity scheduling system developed at Carnegie Mellon University (Sadeh 1994). It employs a search procedure called *micro-opportunistic search* (described in Section 2.3.1) to constructively build high quality just-in-time schedules. Experimental results reported in Sadeh (1994) indicate that Micro-Boss outperforms several other scheduling techniques proposed in the OR and AI literature, including the variety of dispatch rules and coarser granularity (macro-opportunistic) search techniques. Comparison with neighborhood search techniques such as simulated annealing has also shown that Micro-Boss can generate particularly high quality solutions in very little time.

Micro-Boss was initially developed for generalized just-in-time job-shop scheduling problems (Sadeh 1991). Later research by Norman Sadeh and the author has focused on further generalizing the modeling framework to allow for the representation of properties found in a wide range of real-world production environments (described in Section 2.3.2). This work has resulted in a recent reimplementaion of Micro-Boss. This new version has also been designed to better support flexible, accurate, and competitive supply chain coordination. Thanks to these modeling considerations, the fast implementation of a testbed for evaluating the proposed coordination policies described in later chapters of this dissertation has been made possible.

2.3.1 The micro-opportunistic search procedure

Micro-opportunistic search is based on ideas similar to the bottleneck-centered scheduling approach of OPT in that bottleneck decisions should drive other decisions.⁵ The traditional bottleneck-centered procedure basically involves three steps: (1) identifying the bottleneck resources, (2) scheduling the bottleneck resources, and (3) scheduling the remaining activities. However, in many situations this way of problem decomposition is far from optimal. Every scheduling decision

⁵ One of OPT's scheduling ideas is that once the sequence of activities on the bottleneck resource(s) is given, then the remaining schedule will be easy to complete (see page 9 for details).

made during the construction of a schedule will change the conditions for the remaining, unscheduled part of the problem. The micro-opportunistic search procedure is motivated by this dynamic nature of bottlenecks. Each time one or a small set of activities has been scheduled, a new analysis is carried out to determine the most severe bottlenecks. The result of this analysis is then used to determine which scheduling decision to focus on next. This is repeated until a goal state is reached where the whole problem is solved. By keeping the granularity of the search as fine as possible (at the *micro*-level), the focus of attention can *opportunistically* be shifted from one part of the problem space to another as the location and severity of remaining bottlenecks evolves during the construction/revision of a schedule. The steps of the procedure are as follows:

1. Initialize the problem.
2. Apply *constraint/apparent cost propagation*.
3. If a current state is a complete and feasible solution, then exit. Otherwise, if a conflict is found, then apply a *conflict resolution technique* to resolve the conflict and return to step 2.
4. Apply *variable ordering* to identify the next activity to reserve.
5. Apply *value ordering* to identify when and where to reserve the activity.
6. Reserve the activity and return to step 2.

We will now discuss each of these steps in further detail:

Problem initialization

The initialization step involves loading the description of the model and the activities to schedule, and the initialization of values used by the procedure.

Constraint/apparent cost propagation

This step includes incrementally calculating lower and upper bounds for the time when each of the activities can be reserved with respect to the constraints involved in the problem and the scheduling decisions previously taken. A conflict is identified if constraint propagation results in an activity's temporal lower bound being greater than its upper bound, or if a resource is over-allocated within a time interval. This step also involves the determination of *ideal times* and *apparent costs*. If we consider all possible and feasible times between the lower and upper bound where the activity can be reserved, the ideal time is the time that by local (job-centered) considerations meets the objective function best.⁶ Selection of the ideal time does not consider

⁶ It is actually a heuristic approximation, as described in Sadeh (1991).

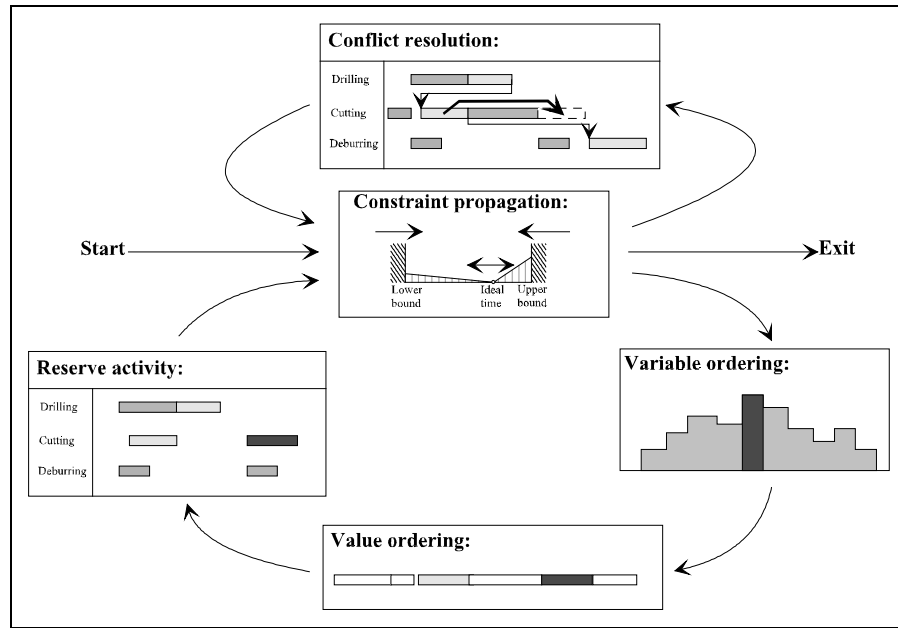


Figure 2: Outline of the micro-opportunistic search procedure.

whether or not reserving the activity at its ideal time will have an undesirable effect for the remaining part of the scheduling problem. For each activity, the *apparent marginal inventory cost* and the *apparent marginal tardiness cost* are maintained. Activities scheduled before their ideal times incur penalties as determined by their apparent marginal inventory costs, while activities scheduled past their ideal time incur penalties as determined by their apparent marginal tardiness costs. These apparent costs are updated as activities are scheduled.

Conflict resolution

A conflict represents a dead-end in the search tree of the problem. By retracting the most recent decision (undoing the reservation of the last activity reserved), the conflict will be resolved and other branches in the tree may be explored.⁷ Backtrack search is NP complete, even though average case complexity reported in Sadeh (1991) was polynomial across a broad range of problems. To guarantee rapid convergence in all possible situations, new conflict resolution mechanisms have more recently been implemented. For example, the box in Figure 2 labeled “Conflict resolution” illustrates a *non-monotonic* technique to resolve a precedence constraint

⁷ The conflict resolution technique initially used in Micro-Boss was *chronological backtracking* (Sadeh 1991).

violation (the arrow from the finish of the second drilling activity to the start of the second cutting activity) by right-shifting the downstream conflicting activity.

Variable ordering

Variable ordering involves determining the resource that has the most severe peak of *resource contention* as well as the activity that contributes the most to this peak. This activity is referred to as the *critical activity*. Resource contention is the result of several activities competing for the same resource around the same time. The determination of resource contention takes a probabilistic approach. If we were able to schedule every activity at its ideal time, we would obtain an optimal solution. This is often not possible due to competition from other activities. The goal is therefore to reserve every activity as close as possible to its ideal time. We assume a probability density function for where the activity will eventually be reserved. This function is distributed between the lower and upper bound of the start, with the ideal time as the most probable value, and biased by the apparent marginal inventory cost and apparent marginal tardiness cost. From this probability density function and the resource availability we can derive a *demand profile* for each activity. The demand profiles are then aggregated on every resource into *resource contention profiles*. These resource contention profiles allow us to search for the most severe peak of demand. Finally, the critical activity is found as the activity that contributes the most to this peak.

Value ordering

Value ordering takes a resource-centered approach to select the time when the critical activity should be reserved. The time selected is the one that minimizes the cost incurred by the job for the critical activity and by other competing jobs. This is equivalent to solving a single-machine or parallel-machine early/tardy problem in which activities scheduled before their ideal times incur earliness penalties and activities scheduled past their ideal times incur lateness penalties. The single-machine early/tardy problem is an NP-complete problem in itself. Micro-Boss therefore relies on several variations of parametric release heuristics and dispatch-based heuristics to generate a number of single- or parallel-machine schedules. Value ordering returns the reservation time for the critical activity from the schedule with the lowest sum of earliness and lateness penalties.

Activity reservation

The final step is to reserve the critical activity. Activities are reserved by updating the state of the corresponding resource intervals.⁸ The scheduled time and the resource intervals with a state change (e.g., change in available capacities) become sources of subsequent (incremental) constraint propagation.

We have now presented the fundamental search procedure of Micro-Boss. While one of the main strengths of micro-opportunistic scheduling is its ability to quickly produce high quality solutions, the basic procedure has been refined over the years and made even more efficient through the introduction of a number of speedup mechanisms (e.g., rough demand profiles (Sadeh 1994)). The current version of Micro-Boss is able to build high quality schedules for problems involving tens of thousands of activities in a matter of minutes (e.g., in the case of a Raytheon machine shop where Micro-Boss was deployed in the summer of 1997, 35,000 activity schedules required about 10 minutes to be generated on a 166 MHz SUN Ultra SPARC 1 with 152 Mb of RAM). More details on the Micro-Boss search procedure and experimental results of the effectiveness of the procedure can be found in Sadeh (1991, 1994).

2.3.2 The Micro-Boss modeling framework

The Micro-Boss modeling framework is designed to overcome the challenges discussed in Section 2.2. The target of the design is a system allowing the most common attributes that are important for the creation and representation of valid and high quality schedules. Since there is always a tradeoff between generality, complexity, and performance, the core system is designed on the basis of a general scheduling *ontology* whereby the assumptions made about the problem at hand are kept to a minimum. A set of low-level modeling primitives is developed for the purpose of allowing higher level scheduling elements (e.g., complex activities under idiosyncratic constraints) to be declared (without recompilation) based on different combinations of the low-level primitives. Complexity is controlled through a modular, object-oriented design, and performance is maintained by allowing generic (and therefore in some cases slow) methods to be overloaded to account for meta-level knowledge about the problem domain.

We will now present some of the aspects of the framework in more detail. First, we will define the high-level ontology used in our scheduling framework. We will then explain how activities, temporal constraints, and resources are modeled.

⁸ What we mean by the “state of a resource interval” will be explained in Section 2.3.2.3.

Finally, we will provide an example of how a given environment can be modeled. Aspects of the modeling framework that relates to supply chain coordination will be presented in Chapter 4.

2.3.2.1 Scheduling ontology

The foundation for the scheduling framework is a scheduling ontology that specifies at a high level what exists in a scheduling model. An ontology specifies a common *terminology* that may be transferable across various scheduling domains (and between individual scheduling agents). The abstract entities of the Micro-Boss ontology are:

- *Context* — the set of assumptions defining a scheduling problem and a complete or partial solution to the problem.
- *Resource* — an entity whose capacity can be allocated to various activities.
- *Process plan* — a description of how to manufacture a particular product, including the steps to be performed, their sequence, the various resources to be involved, and the standards for setup and run. A process plan is therefore a template for the creation of a job.
- *Product* — a commodity or service that can be manufactured/assembled/delivered through the coordinated use of a set of resources according to one or more available process plans.
- *Demand* — a need for a particular product, including quantity, due-date, and priority.
- *Item* — any unique manufactured or purchased part, material, intermediate subassembly, or product.
- *Order* — an order (or production order) is created to meet one or more demands for a particular product (e.g., from a customer order). It comprises a collection of jobs to meet each of these demands.
- *Job* — a collection of activities connected by temporal constraints for the purpose of satisfying a demand.
- *Activity* — each step of a process plan becomes an activity of the real job. An activity requires the availability of a particular resource or set of resources to be performed. It can also be thought of as a placeholder for a set of constraints or as synchronization points for the utilization of one or more resources.
- *Reservation* — the allocation of a resource to an activity over one or more time intervals.
- *Constraint* — a condition that must be met for the schedule to be valid.

Using this terminology, scheduling can be defined as:

The process of feasibly synchronizing the use of
resources by *activities* to satisfy *demands* over time.

2.3.2.2 Modeling activities and constraints

The framework provides activities at two levels of abstraction, which we will call the *activity level* and the *constraint level* (see Figure 3). The activity level corresponds to the traditional view, where each activity includes an identifier and information about its predecessors and successors. The constraint level gives a more detailed representation of activities, representing them as collections of low-level modeling primitives. These modeling primitives are grouped into *nodes* and *constraints*. Nodes represent temporal events and store the values necessary for performing constraint-directed scheduling (including lower bound, upper bound, and scheduled time). Each constraint connects two or more nodes. Together nodes and constraints build a network where temporal information can flow and be stored.

Nodes are divided into sub classes of *fixed* and *free* nodes. Fixed nodes establish anchor points (or reference points) in the network. Free nodes are nodes that allow values to be updated, such as to restore consistency with the constraints. They also keep track of *currently feasible values* by means of constraint propagation.

Each constraint is a channel for information flow between nodes. Thus, within this network, scheduling becomes the process of assigning and maintaining values in all free nodes so that the resulting schedule satisfies the constraints. Constraints are divided into relaxable and hard constraints. A relaxable constraint, for example, the amount of time a job finishes past its due date, can be thought of as measuring the *distance* between the nodes it connects. It does not participate in the

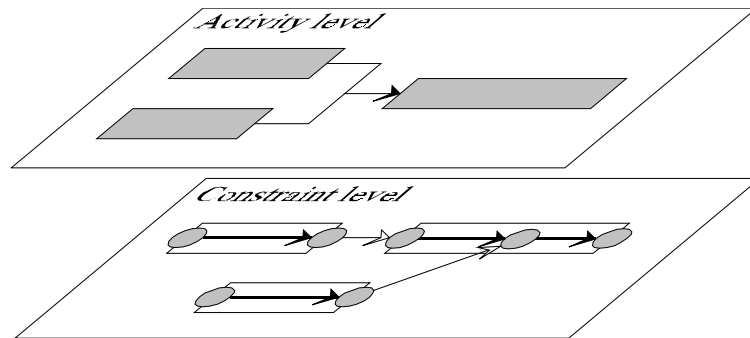


Figure 3: Two levels of abstraction for representing activities and temporal constraints. At the activity level we can see an example of three activities in an assembly-type process plan. The constraint level provides a more detailed view of the activities as nodes (gray circled) and constraints (phases as thick arrows and temporal constraints as thin arrows).

constraint propagation process, but might (dynamically) update some information needed to evaluate the schedule and guide the scheduling process. The hard constraints direct the search during scheduling. They are divided into two sub classes. The first and most obvious is the class of *temporal constraints*, which includes the *minimum-separation-constraint*, the *maximum-separation-constraint*, and the *minimum-maximum-separation-constraint*.⁹ Together they enable the representation of any temporal relation of Allen's interval algebra (Allen 1984). The second is the class of *phases*.¹⁰ A phase is a constraint that spans a period of an activity's duration, and that requires the availability of one or more resources to be satisfied. A given activity may include more than one phase. The framework includes a small but easily extendible library of different types of such phases. These are:

- *resource-driven-duration-phase* — the duration is determined by the quantity to be processed and the efficiency of the resource (e.g., sequential processing of a number of identical items).
- *conditional-duration-phase* — the duration is determined by the state and efficiency of the resource (e.g., sequence dependent setups, tool replacement).
- *minimum-maximum-duration-phase* — the duration is bounded by a lower and upper threshold (e.g., cooling, heat treatment, mixing).

Furthermore, all phases include a flag determining whether or not the execution of the phase can be split across periods of downtime. A multitude of variations of activities can be modeled as combinations of these primitives as long as the following properties are satisfied:

- The set of resources required should not change within the phase.
- The condition for determining the duration of the phase must be unambiguous, for example, whether the phase is conditional or unconditional, whether the phase is splittable or not, or whether the duration is resource-driven or not.

⁹ All these constraints may be obtained from the minimum-separation-constraint alone. The maximum-separation-constraint is a minimum-separation-constraint in reversed direction and the minimum-maximum-separation-constraint is a conjunction of a minimum-separation-constraint and a maximum-separation-constraint. However, they are distinguished both for efficiency reasons and to be more intuitive.

¹⁰ To our best knowledge, this low-level representation of activities into phases has previously only been reported in planning systems, for example, O-Plan (Tate and Drabble 1995).

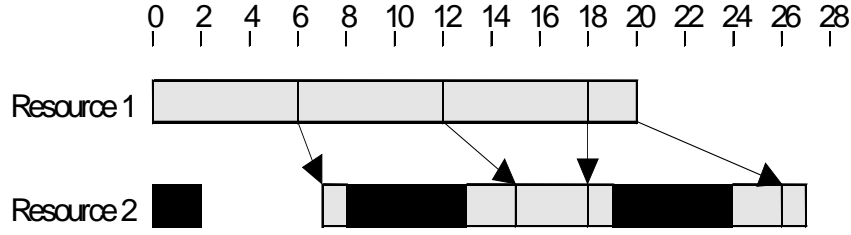


Figure 4: Phased activities for modeling transfer batches. A batch of 100 items is split into 4 transfer batches (i.e., phases), the first three each have 30 items and the last 10 items. Resource 2 is twice as fast as resource 1, but includes downtime. The activities are preemptive. Downtime is shown as black areas and temporal constraints (due to transfer of material) are shown as arrows. In this specific case it is the 3rd temporal constraint that constrains the maximum amount of temporal overlap.

A situation that often occurs in industry is the use of *transfer batches*. In batch-producing environments, it might be undesirable to wait until an entire batch is completed on a resource before the downstream resource can start working on the batch. Time can be saved if the activities can overlap in time. This can be achieved by introducing transfer batches of smaller size than the full batch. In situations where activities are preemptive (i.e., can be interrupted by downtime) and resources are subject to different shift patterns, it might be impossible to predetermine which of the transfer batches will be the most constraining so that start-to-start constraints with a pre-calculated fixed lag to account for this situation can be applied. We can instead model each activity as a sequence of phases, where each phase represents a transfer batch. The flow of material can then be synchronized by introducing a minimum-separation-constraint from the finish of the supplying phase to the start of the consuming phase. An example is shown in Figure 4.

2.3.2.3 Modeling resources

Traditionally, resources are thought of as having a specific capacity that can be allocated for the execution of activities. The Micro-Boss framework suggests a broader view of resources as representing temporal state-variables. When examining a resource R for availability A over a period I of time, we check whether the state S_I of the interval satisfies the conditions C that the phase requires.

$$A(R, I, C) = M(S_I, C)$$

where M is a predicate that encodes the success of matching the state and the condition. In the simplest case, S_I indicates the free capacity of the interval and C is given as $S_I \geq c$, where c is the capacity requested for allocation. Hence, M will return

true if there is sufficient free capacity in the interval to accommodate the request. This general definition of resources and availability checking allows for the definition of a broad range of more specialized (and thus more efficient) resource classes:

- *Unary-resources* can be assigned to at most one activity at a time.
- *Conditional-reservation-resources* are resources that keep track of both *conditional reservations* (e.g., a sequence-dependent setup) as well as the standard reservations.
- *Batch-capacity-resources* allow for cumulative reservations that are compatible and synchronized (e.g., an oven).
- *Multiple-capacity-resources* allow for cumulative reservations as long as the overall capacity is not exceeded. Homogenous pools of identical resources can in some cases be modeled accurately as multiple-capacity-resources.
- *Aggregate-resources* define pools of parallel (homogenous) or alternative (heterogeneous) sub resources.
- *Mobile-resources* are batch-capacity-resources that also keep track of the location of the resource (as part of its state).
- *Replenishable-resources* represent resources for which the capacity required or provided by activities changes the capacity of the resource past the completion of the activity. Required capacity will definitely be absent and provided capacity will definitely be available until another activity consumes it.

Any of these resources may be subject to a resource calendar. Resource calendars specify patterns of availability, such as shift patterns, weekends, and holidays. They can easily be modified for analyzing “what-if” scenarios with respect to resource availability.

The modeling framework also permits the representation of shop-floor status (such as current time, resource breakdowns, presence of raw material, and execution status of individual activities), which is an essential part of the definition of a scheduling problem and of our supply chain coordination experiments. This framework has been tested on problems found in many different settings and has shown to be a powerful means for representing the essential characteristics and idiosyncrasies of these various environments (Sadeh 1995).

2.3.2.4 An example

We will now use the simplified chemical plant shown in Figure 5 as an example to illustrate the power of the modeling framework. The plant works as follows: A blending pump (indicated with a “P” in the figure) can extract and blend raw materials and send the blend into a mixing tank where the blend is mixed. For quality reasons, the mixing process is bounded by a minimum and a maximum duration. The mix is then pumped into a filler station where it is filled into various cans. Since blender pumps and filler stations have no storage capacity, the mixing tank is required from when blending starts until filling has been completed. The duration of blending is proportional to the batch size, and the duration of filling is determined by the batch size and the size of the cans. Blending and filling can only be performed during regular (daytime) hours.

If we were to model each phase of the process as a separate activity, we would face the problem of how to guarantee that the same mixing tank is selected throughout the whole process, and if we model each resource requirement as a separate activity, we cannot determine the duration of “the mixing activity.” Using a phased activity approach facilitates the modeling of this situation. The activity is thereby modeled with three phases, as shown in Figure 6. The 1st and 3rd phases are standard resource-driven-duration-phases while the 2nd phase is a minimum-maximum-duration-phase.

Blending pumps and filler stations are represented as unary-resources (assuming that time for cleaning between consecutive runs of different mixtures is negligible). They are assigned resource calendars to limit their availability to daytime hours. Mixing tanks are represented as batch-capacity-resources that allow

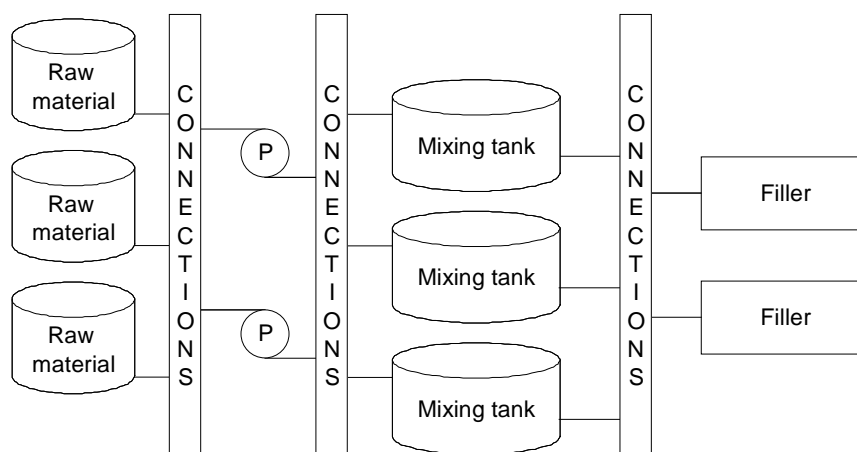


Figure 5: A chemical plant that produces chemical mixtures.

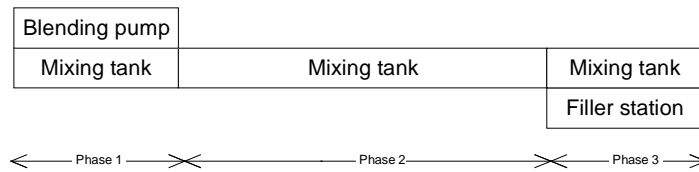


Figure 6: A process in the chemical plant represented as a single activity that is split into phases with different properties. Execution of the first phase requires a blending pump and a mixing tank to be present, the second phase occupies the mixing tank, and the third phase needs both a mixing tank and filler station.

cumulative reservations of identical mixtures. Each of these three pools of alternative resources is represented as an aggregate-resource. The process plans refer to these aggregated resources, allowing the search procedure to allocate a specific resource during scheduling.

3. Supply chain management

This chapter is intended to introduce the reader to supply chain management. Section 3.1 will give a general introduction to the area, including a presentation of some of the associated managerial challenges as well as a review of certain analytical work that motivates the perspective to supply chain management taken in this dissertation. We will then focus on issues that relate to the modeling of distributed frameworks for the coordination of autonomous entities. These topics are covered in the three subsequent sections. Section 3.2 addresses supply chain management from the perspective of distributed artificial intelligence. Section 3.3 discusses aspects of supply chain modeling. Finally, some issues relating to the autonomy of supply chain entities are discussed in Section 3.4.

3.1 Introduction to supply chain management

As they transition towards a global market economy, companies are increasingly focusing on specific, high-value-adding manufacturing niches. Consequently, it is becoming more common to purchase components from suppliers (see, e.g., Womack *et al.* 1990). Furthermore, to remain competitive, companies are constantly faced with challenges to reduce time-to-market, improve product quality, and reduce production costs and leadtimes. These challenges cannot be met effectively only by changes within specific organizational units, but rather critically depend on the relationships and interdependencies between different organizations, both internal and external to a given company.

It is often argued that coordination between suppliers and customers enables a large bottom-line benefit potential for everyone involved, as in this quotation from Harmon (1992):

Purchase constitutes the single greatest potential for improving the quality of our manufactured products while reducing their costs and, consequently, their prices. Materials and purchased components account for 60 to 70 percent of the cost of goods manufactured in almost every company. Each manufacturer's inventory investment and customer service level are thus controlled less by its own processing time than by its suppliers' leadtimes and delivery reliability.

As a result, the welfare of any business entity in the supply chain directly depends on the performance of the others, along with their willingness and ability to coordinate. This is what supply chain management is all about. In supply chain management one can distinguish between strategic, tactical, and operational level decision-making aimed at optimizing supply chain performance. The strategic level defines the supply chain network. Primary products are selected, and supply chain and product design must conform. Setting up the supply chain includes, for example, the selection of possible suppliers, transportation routes, manufacturing facilities, production levels, and warehouses. At the tactical level, process plans and master production schedules are created. The operational level is where detailed schedules are created and executed. Strategic, tactical, and operational level decision-making functions are distributed across the supply chain and may be integrated through *lateral coordination*. Strategic and tactical decision-making often utilize a higher level of abstraction than operational decision-making. For example, manufacturing facilities and transportation routes might be the lowest level of details considered in the process of configuring a supply chain, while the operational level might consider individual machines and labor and transportation units. In fact, as the three traditional levels of decision-making are integrated into a single real-time model, the distinction between them becomes blurred. Tactical decisions such as negotiating orders can be dealt with in real-time, and even the selection of suppliers, which traditionally has been considered a strategic decision, can in some cases become an operational decision.

Supply chain management is integrative in that it spans a wide range of business disciplines and research areas. The literature addresses general trends and overviews of research activities (Thomas and Griffin 1996; Rolstadås 1995; Sadeh and Smith 1993), benchmarking efforts identifying trends and philosophies based on comparative analysis of current practices (Hall 1983; Womack *et al.* 1990; Lyons *et al.* 1990; Helper 1991; Lee and Billington 1992; Davis 1993; Leavy 1994; Knill 1994; Henkoff 1994; Srinivasan *et al.* 1994; Arntzen *et al.* 1995; Choi and Hartley 1996), strategic methodologies (Porter 1980; Cohen and Lee 1988; Lee and Billington 1993), strategic discussions concerning inter-organizational information systems and standards (Benjamin and Wigand 1995; Upton and McAfee 1996), supplier-vendor coordination (Bassok and Akella 1991; Anupindi and Akella 1991; Anupindi 1993; Lau and Lau 1994; van der Duyn Schouten *et al.* 1994; Swaminathan 1996), inventory-distribution coordination (Clark and Scarf 1960; Svoronos and Zipkin 1991), production-distribution coordination (Pyke and Cohen 1993; Anand and Mendelson 1997), operational methods for consistency enforcement across supply chains (Thierry *et al.* 1993; Beck and Fox 1994; Simonis

and Cornelissens 1995), generic supply chain modeling efforts (Cohen and Lee 1988; Swaminathan 1996), and supply chain coordination frameworks (Kalakota *et al.* 1995; Sadeh 1996; Fox and Gruninger 1998).

Lee and Billington (1992) study actual examples in industry in order to provide a well-structured discussion of managerial pitfalls and corresponding opportunities in strategic supply chain management. Below we will excerpt each of the strategic pitfalls identified by Lee and Billington in their study since they all are significant to enable good operational performance of a supply chain. Pitfalls 6, 7, 9, 11 and 12 will not be further addressed in this dissertation.

1. No supply chain metrics — Although the supply chain's overall performance depends on the joint performance of various sites, each site is usually managed rather autonomously, and the objectives of each may conflict with the supply chain's overall performance. A common dilemma is the position of inventory buffers. For example, local performance may be improved by cutting inventories even though the overall supply chain as measured by customer satisfaction might thereby suffer. Performance metrics should, therefore, include measurements for the complete supply chain as well as local measurements.
2. Inadequate definition of customer service — A supply chain will ultimately be measured by its responsiveness to customers. However, there are different definitions of responsive customer service, such as percentage of items shipped prior to customer due dates, percentage of completed orders shipped prior to customer due dates, degree of order lateness, leadtime, and response time. The appropriateness of each of these depends on the circumstances. For example, if an order is merely to replenish stock, the supplier can send individual items separately. If the customer requires the complete order before the supplier can complete a job, the fill rate should be measured in terms of completed orders.
3. Inadequate delivery status data — When customers place orders, they want to know when their products will arrive. Many companies publish their standard response times although these might not resemble actual response times, and they might not be able to revise shipment dates as changes are introduced during the order cycle.
4. Inefficient information systems — Information retrieval may become a tedious manual process if the different information systems are not linked and suppliers cannot quickly and systematically retrieve the information they need to set shipment dates. This makes it impossible to quote accurate shipment dates, as discussed in pitfall 3, and also discourages short production planning cycles, resulting in increased forecast errors.

5. Ignoring the impact of uncertainties — There are many sources of uncertainty in a supply chain. In order to reduce the impact of these uncertainties, supply chain managers must understand their sources and the magnitude of their impact, and then work on ways to reduce or eliminate them. For example, before starting to modify the inventory stocking policies for purchased parts to avoid stockouts, they should concentrate on ways to improve the suppliers' delivery performance, one of the root causes of the problem.
6. Simplistic inventory stocking policies — Stocking policies should differentiate between parts (depending on reliability of supplier, stability in demand, and criticality of part) and periodically be adjusted to reflect changes.
7. Discrimination against internal customers — A division or entity in an organization may produce items for a mix of internal and external customers. Customer service for internal customers may not always be tracked, while external customers bring in real revenues and thus are more visible and apparently more valuable. This prioritization directs the effects of uncertainty to the internal customer and can hurt the company's overall profitability.
8. Poor coordination — In the absence of good coordination, a supplier may be forced to operate with high finished goods inventories. These higher inventory levels translate into costs that eventually find their way back to the customer. The supplier might also have to expedite deliveries using costly means of transportation.
9. Incomplete shipment method analysis — The tradeoff between inventory and shipment costs should be adequately analyzed. For example, cutting shipment costs by utilizing slower or less frequent means of transportation might increase inventory investments both in the transportation pipeline and in safety stocks.
10. Incorrect assessment of inventory costs — Estimation of inventory costs should consider lost opportunity costs, warehousing and storage costs, and obsolescence costs (due to short life cycles and rework to meet engineering changes).
11. Organizational barriers — A company may have a decentralized organizational structure, each organization having its own performance measures. Sometimes entities within a supply chain belong to different organizations within the same company. The barriers might result in unwillingness to commit resources to help someone else. The supplier might have an unreliable delivery performance, forcing the customer to keep a high level of finished goods inventory. However, the overall inventory investment in the supply chain could be reduced if the supplier was willing to reduce the cycle time or change the level of inventory of semi-finished or finished goods.

12. Product-process design without supply chain considerations — A product can be designed in such a way that customer-specific components can be added at the final stage, or even by the customer. The fluctuations in demand for generic semi-finished products are lower than for individual customer-specific products. If at all possible, delayed product differentiation and component commonality may improve the level of customer service and also reduce costs and leadtimes.
13. Separation of supply chain design from operational decisions — Decisions to open or close a plant or distribution center are often based on fixed costs and transportation cost considerations. The effect of network change on operational efficiency factors, such as inventory investments and order response time, should also be considered.
14. Incomplete supply chains — Going beyond the internal supply chain by including external suppliers and customers often opens up new opportunities for improving internal operations. By better understanding the “customer of the customer’s” inventory control system, internal service targets can be set that better reflect the supply chain’s overall priorities.

A strategic issue that relates to Lee and Billington’s pitfalls 4 and 8 is the investment in and use of information technology as a means for sharing information between suppliers and customers. Srinivasan *et al.* (1994) analyze shipment data in the American automobile industry to determine the impact that the sharing of information and the use of EDI technology have on the level of customer service. Their analysis concludes that the sharing of JIT schedules yields significant reductions in the level of shipment discrepancies and, furthermore, that establishing integrated EDI links for the sharing of information is a source for further improvement, especially when part variety is high.

Another strategic issue relating to the selection of the coordination structure (CS) of a supply chain has been analyzed by Malone (1987) and Anand and Mendelson (1997). Malone describes four generic coordination structures from an organizational point of view and analyzes tradeoffs among them in terms of *production costs*, *coordination costs*, and *vulnerability costs*. These four CS are *product hierarchies* (a separate division per product line), *functional hierarchies* (structuring the organization along functional departments, each responsible for tasks of a certain type), *decentralized markets* (all buyers are in contact with all possible suppliers and decisions about what transactions to accept are local), and *centralized markets* (e.g., the presence of brokers between buyers and suppliers, each responsible for the coordination of tasks of a certain type). Production costs include the costs of production capacity and the costs of delays in processing tasks. Coordination costs

include the costs of maintaining communication links and the costs of exchanging messages along these links. Vulnerability costs are the unavoidable costs of a changed situation that are incurred before the organization can adapt to a new situation, modeled as the expected costs that result when an organization fails to perform tasks. The analysis identifies advantages and disadvantages for each of the CS. In particular, decentralized markets are found effective with respect to production and vulnerability costs but involve high coordination costs. In light of recent developments in information technology, however, there is reason to believe that coordination costs will be less of an issue in the future, and consequently the trend will be towards increased outsourcing and the establishment of virtual enterprises consisting of smaller autonomous entities.

Anand and Mendelson (1997) analyze the effects of different CS on the performance of a firm that sells its product in several independent horizontal markets, each subject to demand uncertainty. The alternative CS are: (1) a centralized CS, where the center makes all the decisions using all the data, but none of the local knowledge, (2) a decentralized CS, where each branch is responsible for its own sales decisions based on local data and knowledge, and (3) a fully distributed CS, where all data are shared and hence each branch makes its decisions based on both its own local knowledge and shared data. Their analysis concludes that for a large number of markets the distributed CS performs better than the decentralized CS, and that the decentralized CS dominates the centralized CS in spite of the superior coordination of the latter. The analysis indicates that the centralized CS, being similar to the approach taken in many of today's ERP-based supply chain coordination systems, will suffer from the lack of ability to make use of the more detailed localized data. It also suggests that a supply chain, as a general rule, will perform best by allowing decisions to be distributed and based on a combination of detailed local knowledge and shared coordination knowledge.

The final strategic issue we will discuss is the selection of supply chain partners. Research on partner selection is based on two dominant perspectives that we will call the *competitive perspective* and the *cooperative perspective*. The competitive perspective in the industrial economics and competitive strategy literature views the customer-supplier relationship in terms of both parties competing with each other for profit margin. In Porter's five force model (Porter 1980), five competitive forces determine the long-run profit potential of any market and its participants: the threat of new entrants, the threat of substitute products, the bargaining power of suppliers, the bargaining power of customers, and the intensity of rivalry among the core competitors. Each of these forces when strong will tend to drive profit out of the market. In this perspective, any company competes in a win-

lose game with its suppliers and customers for the profit margin, and bargaining power is kept strong through the following strategies:

- Maintain multiple sources of supply to avoid reliance on a single supplier. Supplier prices are controlled by constantly searching for alternate suppliers to compete for each delivery.
- Maintain the ability to easily switch between suppliers, for example, avoid long-term commitments to individual supplier.
- Maintain a credible threat for suppliers to bring component production in-house.
- Create and maintain a unique position in the relationship to customers through product, service, or quality distinctiveness.
- Maintain a credible threat of forward integration to keep the customer in-line.

The cooperative perspective presents a very different view, viewing the relationship in terms of a partnership-in-profit creation. According to the competitive model, such a strong dependency on suppliers would weaken the company's bargaining power. However, the perspective arose as companies shifting to JIT production realized their fundamental need for suppliers that were reliable in time and quality. Furthermore, the no-stock policy of JIT exposes a company to fluctuations in customer demands, thus making it more sensitive to customer demand forecast errors. These are the main reasons why JIT-producing companies had to abandon the more traditional competitive perspective in favor of cooperative and long-term relationships with suppliers and customers. The benefits of tight coordination are now well recognized:

- Closer coordination in schedules.
- Cooperation in process and product improvements.
- Joint actions aimed at cost reduction.

These features help to reduce inventory investments and improve profit margins while increasing the overall levels of quality and service deliverable by the partnership to its customers. The partnership protects the supplier from competition in the supply segment and the customer can enjoy many of the benefits of vertical integration, such as greater security of supply and more control over cost and quality. Altogether this becomes a win-win situation for both parties, with equitable shares of the rewards.

Despite all these benefits, the dangers of cooperative supply chain partnerships also exist. One major danger is the risk of selecting the wrong partner.

The risk for the supplier is that he commit a large portion of his capacity to a company that may be unable to hold or strengthen its market position as the industry evolves. On the other hand, the risk for the customer is that he invest heavily in a partnership in which the supplier proves incapable of developing in line with him and ultimately undermines his competitive position. Further discussion of the advantages and disadvantages of the two perspectives from a management point of view can be found in (Lyons *et al.* 1990; Leavy 1994).

There are generally two main drivers for technological innovations, *demand pull* and *technology push*. So far we have addressed demand pull, that is, the obvious need for better information technology in supply chain management. On the other hand, the technology push arising from advances in low-cost information and communication technology is also an important driver. These developments have facilitated the search for prospective suppliers and customers (e.g., by means of electronic marketplaces) and the establishment of short-term contracts (with corresponding communication links for exchange of information during the execution of the contract). The competitive perspective might thus become more and more attractive with this new technology, even for some JIT producing companies, since it allows for reliability through the synchronization of schedules and avoids the dangers of long-term partnerships. However, one challenge that must be overcome is how to guarantee that short-term partners are willing to cooperate and exchange honest information. These issues have been subject to considerable research in the distributed AI and game theory communities (Rosenschein and Zlotkin 1994; Gmytrasiewicz and Durfee 1993; Sen and Durfee 1994; Sandholm 1996), and they will be further discussed in Section 3.4.

3.2 Distributed AI for supply chain management

A supply chain, whether it is internal to an organization or spans across multiple organizations, is usually subject to decentralized control. Suppliers may arrive while others disappear. Each supply chain entity typically has only an incomplete view of the state and operation of other entities. Information is exchanged in real time or periodically through message passing. Since supply chain coordination is fundamentally concerned with coherence (how well the system behaves as a unit) among multiple loosely coupled decision-makers, it is natural to implement supply chain software using agent technology.

Distributed AI (DAI) techniques have been found useful in many applications related to manufacturing, such as the cooperative problem solving (CPS) techniques for scheduling introduced in Section 2.1. These techniques are able to partition the

scheduling problem among distributed agents, thereby being candidate approaches for operational supply chain coordination. However, systems based on CPS techniques would be hard to introduce in inter-organizational environments since they do not support distributed autonomy. We would like to advocate and emphasize the importance of autonomy among supply chain entities:

Any entity in the supply chain has its set of suppliers and customers, which again have their own sets of suppliers and customers. Resources within an entity are normally not dedicated to a given customer. Supply chains thereby overlap (i.e., share resources) with other supply chains. A car parts manufacturer (e.g., producing airbags) might thus be a supplier for both Ford and Chrysler.¹¹ It would in general be impossible to optimize the manufacturer's schedule for both customers' objectives at the same time. The parts manufacturer has his own objectives and is responsible for building his own schedules. The same example also illustrates how difficult it would be to build schedules across supply chains in a centralized way since supply chains are inter-dependent. Our parts manufacturer would have to build his schedules concurrently with the scheduling systems at both Ford and Chrysler, which obviously is impossible. We claim, therefore, that a workable supply chain coordination mechanism has to support an asynchronous mode of schedule revision where contingent inconsistencies between entities are allowed to exist for short periods of time (though one should strive to eliminate them).

A stream of research originating from research in DAI, known as Multi-Agent Systems (MAS), emerged around 1980. Problem solving in MAS is performed by means of communication between a set of loosely coupled *autonomous agents*. Agents in MAS are *goal-oriented*, and they try to attain local goals in complex, dynamic environments. An agent can sense the environment through its sensors and act upon the environment using its actuators. Such an agent is called *autonomous* if it operates completely autonomously, that is, if it decides how to relate the sensor data to motor commands in such a way that its goals are attended to successfully. An agent is *adaptive* if it is able to improve over time, that is, if the agent becomes better at achieving its goals with experience, and if it is able to adjust to a changing environment, such as by dynamically recognizing and making use of new actors in the agent community. Introductions and general overviews of research in the field of autonomous agents are given in (Bond and Gasser 1988; Durfee *et al.* 1989; Chaib-draa 1992; Maes 1994; Genesereth and Ketchpel 1994; Wooldridge and Jennings 1995; Sycara *et al.* 1996; Nwana 1996).

¹¹ These two corporations are chosen purely for illustrative purposes, and because most readers are familiar with their products.

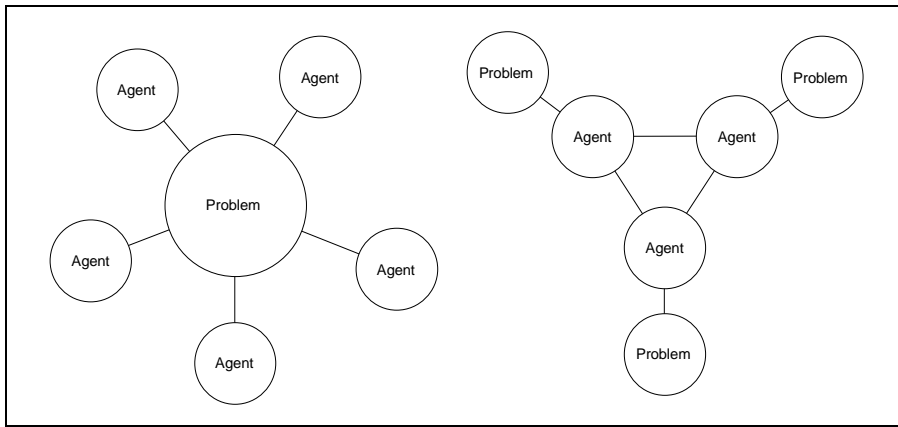


Figure 7: Distributed Problem Solving (left) vs. Multi-Agent Systems (right). In DPS, the problem solving efforts are decomposed and distributed between cooperative agents having different responsibilities. In MAS, individual and loosely coupled problems are coordinated through interaction between goal-oriented agents.

Figure 7 depicts a principal difference between problem solving in CPS and MAS. In a supply chain setting, the “problem” of CPS would be how to create a schedule across a supply chain. An agent participating in several supply chains would, therefore, be involved in solving several problems, one for each supply chain, and would have to make sure that constraints (i.e., capacity constraints) imposed by one problem are taken into account in the process of solving other problems. It would be virtually impossible to resolve these conflicts according to local objectives. On the other hand, problems in the MAS model are solved based purely on local objectives, and conflicts arising between solutions of individual agents are resolved via negotiation.

One focus of this dissertation concerns the coordination of distributed supply chain entities. The coordination of agents has been a subject of extensive research in the distributed AI community. However, these discussions either tend to be extremely general, or hardly seem applicable to supply chain coordination. The research in question can be characterized by various dimensions, including the following:

- Degree of autonomy — the extent to which individual agents have the authority to respond, react, or develop independently of others.
- Degree of cooperation — the extent to which individual agents can be expected to behave benevolently, ranging from full cooperation to antagonism.
- Degree of commitment — the extent to which individual agents are expected to commit to individual decisions, ranging from no commitment, through partial

commitment and the ability to de-commit from earlier commitments, to full commitment contracts.

- Degree of communication — the extent to which coordination relies on explicit communication. On one end of the scale we find systems coordinating by implicit communication, such as behavior-based coordination. On the other end we find systems based on explicit communication, such as negotiation-based coordination. See, for example, Tharumarajah and Bemelman (1997) for a discussion.
- Interaction domain — Rosenschein and Zlotkin (1994) classify interaction domains into task, state, and worth oriented domains. Task-oriented domains define an agent's activity in terms of a set of tasks it must carry out. Agents in state-oriented domains are concerned with moving from an initial state to a goal state. Worth-oriented domains extend state-oriented domains by allowing agents to assign a worth to every potential state.
- Social abilities — the extent to which individual agents are able to assess their surroundings in order to update their model, and to reason about the actions and plans of other agents in order to predict their behavior.
- Structure of the interaction — the extent to which interaction is governed by rules that the agents must follow.

In this space we define supply chain coordination as applying to fully autonomous agents that are expected to cooperate as long as this governs their self-interest. Allocation of tasks is carried out in a task-oriented domain by means of negotiation-based communication. Such communication proceeds according to negotiation protocols, where the resulting contracts ensure a sufficient level of commitment from the agents involved. For example, coordinating schedules during the execution of tasks may be conducted in the worth-oriented domain, where goals can be relaxed to allow for compromises that may lead to increased overall efficiency. Communication is performed by means of message passing, and where the receiving agents may use this information as part of their model of the surrounding world to update its internal schedule accordingly.

3.3 Supply chain modeling

A model may be defined as an abstract representation of a physical system that is used to perform some kind of *off-line* analysis of the system. Supply chain models may, for instance, be created as part of a supply chain reengineering process, or as tools to help answer strategic, tactical, or operational aspects of the supply chain. However, the computer-based information and communication infrastructure of a

physical supply chain may also be viewed as a model in which physical quantities, shipment dates, etc., are represented as numbers. This model is used for *on-line* information processing. In order to distinguish between the two classes of models, we will call the on-line version a *supply chain architecture*. A supply chain architecture is thus a framework for operational supply chain coordination, and designing a supply chain architecture therefore involves considerations of how it is going to operate on a day-to-day basis. A supply chain model, on the other hand, may simply be created to answer a specific question.

3.3.1 Supply chain modeling frameworks

A supply chain model is primarily created for some subsequent analysis of the supply chain. Unfortunately, it is extremely difficult, if not impossible, to create a model that is usable for all types of analysis techniques. A model for mathematical analysis is very different from a simulation model, which again is different from a model for finite capacity scheduling. However, it is possible to create reusable supply chain modeling frameworks for a given analysis technique. They are reusable in the sense that they facilitate analysis for a wide range of supply chain configurations. We will now briefly survey two generic modeling frameworks suggested in the supply chain management literature.

Cohen and Lee (1988) present a prototype for a modular modeling framework that supports analysis of cost/service/flexibility tradeoffs in production/distribution systems for various material management strategies. The framework consists of submodels for material control, production, stockpile inventory, and distribution. The analysis is decomposed so that each sub model is optimized subject to some “local” service target. These service targets also serve as linkages between the submodels, for example, the fill rate from a submodel is used as the service performance measure at the downstream submodel. The advantage of this modeling framework is its ability to perform analysis for a fully integrated supply chain model, but many simplifying assumptions can be found in each model. For example, the production submodel assumes batch manufacturing processing in parallel line flow shops with fixed batch sizes in order to achieve mathematical tractability. Its usefulness for a wide range of applications therefore depends on considerable enhancements in the generality of each module. Cohen and Lee draw similar conclusions based on their experience with the prototype.

Swaminathan (1996) presents a multiagent framework for modeling the dynamics of supply chains by means of simulation. The supply chain is defined as a set of *structural* elements and *control* elements. Structural elements are used to model supply chain entities (retailers, distribution centers, manufacturers, suppliers,

transportation vehicles, etc.). Control elements are used to specify various control policies that govern product flow within the supply chain. A library of generic structural and control elements is given. A supply chain model is constructed by instantiating and relating appropriate structural and control elements. The model obtained may be used to address issues related to *configuration*, *coordination*, and *contracts*, where configuration deal with the network structure of the supply chain, coordination deals with such routine activities as material flow, distribution, inventory control, and information exchange, and contracts control material flow over a longer horizon based on factors such as supplier reliability, number of suppliers, quantity discounts, demand forecast mechanisms, and flexibility to change commitments. A proprietary software application based on some of these concepts has been developed at IBM.

3.3.2 Supply chain architectures

The traditional information systems found in most companies today store information in large databases and periodically download this information between computers as batch processes. However, people have increasingly come to realize that it is preferable to follow events as they unfold and provide their services based on the updated states. This requires their information systems to take an event-driven approach. The challenges these new information systems face are the real time exchange of coordination information between distributed entities as well as the ability to adapt dynamically to changes in the system configuration. Recent academic literature includes several proposals for architectures for such information systems. The manufacturing system literature has proposed architectures based on such ideas as real-time shop-floor control systems (e.g., Lin and Solberg 1992; Cantamessa 1997) and holonic manufacturing systems (Tönshoff *et al.* 1995). These system architectures are conceptually applicable to distributed supply chain systems, but in practice they have only been experimented with at the shop-floor level. We will, however, focus our survey on some of the recently proposed information architectures aimed at supply chain and (virtual) enterprise integration.

The Center for Research in Electronic Commerce, Austin, Texas, is examining a real-time supply chain information system characterized as *an organization of software agents* (Kalakota *et al.* 1995; Hinkkanen *et al.* 1997). For every decision variable in the mathematical OR formulation of the planning model, such as the inventory level for a part, there is an agent in the real-time model that is responsible for the inventory level of that part. For every shared resource there is a coordination agent that allocates the resource among the competing agents. These agents correspond to the constraints in the mathematical model of the supply chain.

There is also an agent for each product in the supply chain. This agent attempts to optimize the flow for the product for which it is responsible. The advantage of assigning agents to every sub problem is that the task for each agent thereby becomes manageable. However, since each agent is myopic and only acts locally, and since the sub problems of the low-level agents are normally strongly interdependent, the challenge is how to coordinate the agents in order to meet the goal of the system in which the agent exists. The approach taken by Kalakota *et al.* is to formalize the relations between agents so that they resemble human organization (hence the notion of organization of agents), namely, certain agents (supervisor or coordinator agents) have authoritative power over other agents and may delegate specific tasks to them. It appears to us that this hierarchical control structure defeats the whole purpose of distributing the control to obtain real-time reactivity. For example, before a decision is taken locally, it must be approved by a coordination agent at a higher level in the hierarchy, which may consult an electronic broker (e.g., an on-line agent capable of solving optimization problems by means of OR techniques) to resolve the problem. Such coordination requires vast amounts of message passing, which in dynamic environments is likely to introduce communication bandwidth problems and more delays than Kalakota *et al.* foresee.

The Enterprise Integration Laboratory in Toronto, Canada, is investigating models of enterprise and supply chain integration (Beck and Fox 1994; Barbuceanu and Fox 1994; Fox and Gruninger 1994, 1998). Their enterprise information architecture is composed of a set of cooperating agents. *Functional agents* are responsible for the planning and control activities in the supply chain. There are functional agents for logistics, order acquisition, transportation management, resource management, scheduling, and dispatching. These functional agents are constraint-based problem solvers who communicate by posting new constraints that must be satisfied. Coordination occurs when agents satisfy not only their own internal constraints but also the constraints of other agents. Negotiation occurs when constraints that cannot be satisfied are modified by the subset of agents involved. *Information agents* support the functional agents by providing information and communication services. The architecture is hierarchical in the sense that the logistics agent defines the constraints for the scheduling agents, and a scheduling agent defines constraints for the dispatching agent. The negotiation process among agents takes a mediated approach, where the functional agent at the higher level acts as the mediator for lower-level negotiation.¹² The advantage of this mediated approach is the improved quality of the solutions provided (as long as agents are

¹² The approach of using a mediator for coordinating agents in distributed manufacturing systems is also adopted by Maturana and Norrie (1995).

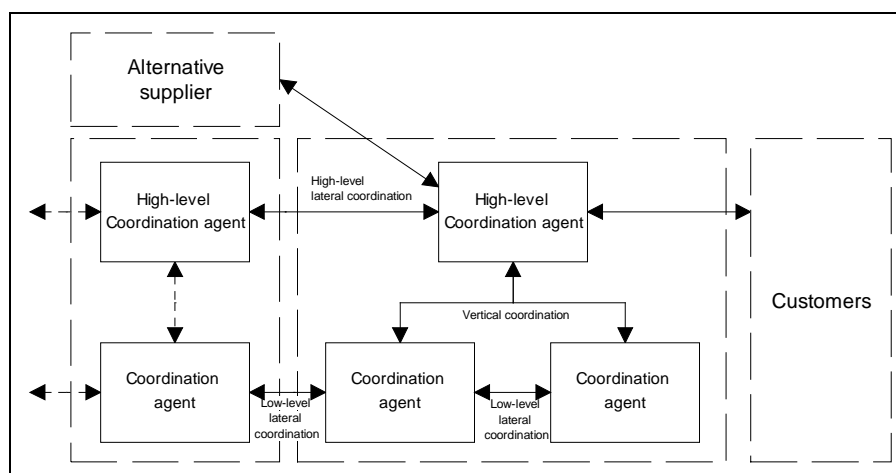


Figure 8: The MASCOT architecture for supply chain management. The dashed lines are organizational boundaries within the supply chain.

working towards a shared objective). However, different coordination mechanisms should be investigated for distributed supply chains of autonomous and self-interested entities.

The Intelligent Coordination and Logistics Laboratory of CMU has proposed an architecture called MASCOT (Multi-agent supply chain coordination tool) for coordination across supply chains (Sadeh 1996). The architecture is shown in Figure 8. Each coordination agent is an extension of the agent-based IP3S architecture (see page 15) to support event-driven coordination and mixed-initiative planning and scheduling decision support functionalities across the supply chain. The agents are organized in different levels of abstraction within each organization of the supply chain, where the low-level agents support single facilities over short to medium term horizon decisions, and higher-level agents are responsible for strategic and tactical decisions across the multiple facilities of an organization. Coordination is performed both laterally and vertically in the supply chain. Lateral coordination protocols support interaction between peer-agents, and vertical coordination protocols support interaction between higher-level agents and those agents that sit underneath them. They thus allow for strategic planning decisions (e.g., integration with product design, where one needs to evaluate alternate design options and associated supply chain arrangements) to be based on updated capacity and material requirements across the supply chain.

The MASCOT architecture has until now been implemented or tested only for a single facility (Sadeh *et al.* 1998). The coordination aspects of this architecture have thus not yet been explored. However, the work carried out in this dissertation

addresses policies for low-level lateral coordination and exchange of available-to-promise (ATP) information that are applicable to the MASCOT architecture.

3.4 Aspects of autonomy

We have argued for the necessity of viewing a supply chain as a collection of autonomous entities. However, this introduces a number of aspects that normally can be neglected under cooperative assumptions. These aspects, which have been addressed in, for example, bargaining theory (Nash 1951) and multi-agent systems (Kraus et al. 1995), will be discussed in this section.

3.4.1 Stability

The stability of a system of autonomous agents relates to whether or not the coordination of the agents has an equilibrium point. A stable system can be characterized as a system in which individual agents remain present (because they benefit from participating) and where their strategies are consistent (e.g., do not change from one day to the next).

Individual rationality

Autonomous agents require individual rationality to participate in the coordinated society. They will only participate (and follow the rules of the society) if they believe that it governs their own interest. Participation is individually rational for an agent if the payoff obtained is not less than the payoff for not participating. Only systems where participation is individually rational are viable in the long run (though, clearly, viability is not something that could easily be determined up front).

Nash equilibrium

Often an agent's coordination strategy depends on what strategy the other agents choose. In such an environment, the Nash equilibrium (Nash 1951) is a fundamental stability criterion. A system is in Nash equilibrium if each agent has chosen the strategy that is the best response to the strategies of all the other agents. Nash proves that every *finite game* has such an equilibrium. However, there are problems in applying Nash equilibrium to a system of autonomous supply chain entities. First, a supply chain network cannot be considered to have a finite number of actors. Furthermore, the characteristics of the participants may change over time, for example, due to technological innovations. There is thus reason to believe that Nash

equilibrium can only be achieved for shorter periods of time, and that the participants should dynamically revise their strategies as conditions change.

3.4.2 Optimality

When each participant in a system has selected his strategy, we would like to know whether this makes the system behave optimally or not. Defining optimality is not a trivial matter even for systems with a single agent, and it becomes even harder to define for a collection of autonomous agents.

Social welfare

Social welfare measures the system-wide good of the agents and is the sum of all agents' utilities. It has restrictive use for autonomous agents since it requires an inter-agent utility comparison (how can we compare the utility of a company that tries to maximize its short-term profit with one that tries to maximize its customer service?). A measurement of social welfare may, therefore, often have to be based on approximations (or projections) of individual utilities. This concept is similar to that of establishing long-term, mutually beneficial supply chain relationships.

Pareto optimality

Pareto optimality describes a solution that cannot be improved upon for one agent without lowering some other agent's utility (Rosenschein and Zlotkin 1994). Pareto optimality hence acquires a global perspective, but without having to compare individual utility functions. Solutions that maximize the social welfare are a subset of the pareto optimal solutions. Once the sum of all agents' utilities is maximized, an agent's utility can only be improved by lowering another agent's utility. Reaching a pareto optimal solution among supply chain agent will, for example, require each agent to provide information that other agents can use to improve their local performance without deteriorating the performance of others.

3.4.3 Fairness

The aspect of fairness is related to both the stability and optimality of a system. An entity may only be willing to participate in systems if it considers the coordination to be impartial (or advantageous to the entity). Furthermore, the optimal behavior of a system may only be obtained when there is a fair competition among the entities involved.

Distributed decision-making

The decision-making process should be distributed. An individual entity should be able to participate in decisions that govern his own interest. There should, therefore, be no central agent that manages the process and makes decisions on behalf of other entities.

Symmetry

A coordination mechanism should not treat agents differently because of non-relevant attributes. For example, preferences should not be based on factors such as the alphabetical order of names if customer service is the true criterion for competition. Furthermore, agents who are exactly alike should receive the same payoff over time. Symmetry is especially important in environments using automated negotiation.

Computational efficiency

The speed with which a system converges to solutions is an important issue in real-time environments whether distributed among autonomous entities or centrally controlled. In systems of autonomous entities, however, it is also important to either seek a fair distribution of the computational load, or to factor the cost of computation into the entity's payoff.

3.4.4 Negotiation

Smith and Davis (1988) identify *task sharing* and *result sharing* as the two basic types of information sharing in distributed problem-solving. Task sharing is exemplified in the contract net, where the agents opportunistically decompose tasks and share work by means of negotiation to get subtasks done. Under result sharing, each node works on some aspect of a problem and shares portions of its partial results with the other agents.

Extensive research within multiagent systems has addressed issues of automated negotiation between agents. One of the original research efforts, and probably the most referenced, is the Contract Net Protocol (CNP) (Smith 1980; Davis and Smith 1983; Smith and Davis 1988). CNP provides a bidding mechanism to allocate tasks to the best suited agents among alternative agents. The net consists of a set of nodes that negotiate with one another by the passing of messages. Each node in the net may dynamically take the role of a *manager* or a *contractor*. A manager is responsible for monitoring the execution of a task and processing the result of its execution. A contractor is responsible for the actual execution of the task.

Typically, a node will take both roles, often simultaneously, for different contracts. Each contract includes the following steps:

1. Task announcement — the manager of a task initiates contract negotiation by advertising the existence of the task to other nodes.
2. Task announcement processing — each node maintains a ranked list of announcements that have been received and have not yet expired. New tasks are inserted in the list if the node is eligible to bid on the task.
3. Bidding — the node selects a task from its list of task announcements on which to submit a bid. The bid includes a specification of the capabilities of the node that are relevant to the announced task.
4. Bid processing — as soon as the manager receives a bid that is satisfactory, that bid is awarded to the associated bidder, which then becomes a contractor for the task.

Extensions of CNP based on micro-economic principles have later been proposed. Sandholm (1996) analyzes negotiation issues for self-interested agents, that is, agents who negotiate and execute contracts as a means to maximize local payoffs, without concern for the global good. These assumptions are appealing in the context of supply chain coordination. His work is based on automated contracting within the CNP framework, and it extends the latter to self-interested computationally limited agents and to contract execution. Within Sandholm's framework, agents pay each other as a reward for handling tasks. Contracts are made on the basis of local marginal cost calculations, that is, whenever a contractor is able to carry out the task at lower cost than the manager agent. Furthermore, Sandholm proposes a leveled commitment contracting protocol (as opposed to full commitment contracts), allowing agents to de-commit to contracts, and also protocols for engaging in multiple negotiations simultaneously (starting negotiations while other bids are pending). Sandholm's protocol has been implemented and tested in a distributed vehicle routing application. There is, however, reason to believe that the protocol can be applied successfully for negotiating contracts between suppliers and customers in a supply chain.

3.4.5 Veracity

For systems of fully cooperative agents, an implicit assumption is normally that agents behave truthfully. This assumption is not valid in systems of self-interested agents. It would be naive to assume that self-interested agents necessarily provide sincere information. A self-interested agent may behave insincerely whenever he

benefits from doing so, a type of behavior that may result in lies or the withholding of information.

One interesting aspect of negotiating contracts arises in situations where the customer wants each supplier to bid what he considers to be the *true* value. This might, for instance, be the case in the following example. A customer announces to a set of prospective suppliers a request for bid for producing a product. The price he is willing to pay is given (and non-negotiable). The request also includes a penalty cost for delivering the product past the promised date, for example, the customer will pay nothing if the product is delivered late. In the traditional auction method (sealed bids, the best bid being granted according to its terms), each bidder will adjust the promised delivery date in the bid based on more and less random speculations regarding its competitors. For example, if the bidder believes that he is able to deliver the part by Tuesday, and that no competitor is able to deliver until Friday, then he might adjust the promised delivery date to Thursday.

Vickrey (1961) addresses this kind of scenario and proposes an alternative auction method whereby the best bid is granted according to the terms of the second best bid. It is argued that this auction method, called the *Vickrey auction*, makes each supplier bid his *true* date. The argumentation is as follows: Bidding a later date than the *true* date could only diminish a supplier's chances of winning at what would have been a profitable date and could not affect the date of the granted contract if he were the successful bidder. Bidding an earlier date than the *true* date, on the other hand, would increase his chances of winning, but only under circumstances that would involve him in a transaction expected to be unprofitable. The truthfulness obtained from the Vickrey auction is not free but rather comes at the expense of "paying" the difference between the best and the second best bid. However, adding mechanisms such as the Vickrey auction to the negotiation process might be necessary to ensure stability of the coordination policy. A line of research in *game theory* (Rosenschein and Zlotkin 1994) has expended considerable effort in investigating appropriate policies whereby the optimal strategy of each agent would lead to the pareto optimal (and stable) performance of the system, as exemplified by the above discussion regarding Vickrey auctions.

Gmytrasiewicz and Durfee (1993) provide a study of interaction between agents based on the assumption that communicative behavior is guided by economic rationality, illustrating how honesty and truth can emerge even from rational and self-interested behavior. Their assumption is that agents transmit messages in order to increase the effectiveness of interaction as measured by their expected utilities. Their analytical study concludes that agents who communicate repetitively will tend to tell the truth to each other and to trust each other, while "outsiders" will likely be lied to

and not believed. This conclusion is interesting in light of our discussion in Section 3.1 regarding the selection of supply chain partners since it provides an argument for close partnerships. We will not go further into the aspects of truthfulness of interaction but simply assume that Gmytrasiewicz and Durfee's conclusions apply. Information received from established supply chain partners is assumed to be truthful (until the opposite is proven) while external customers must be dealt with more carefully, for example, by means of negotiation mechanisms based on Vickrey auctions.

4. A framework for supply chain coordination

This chapter presents a framework for operational coordination of a supply chain. The framework is a subset of the MASCOT architecture presented in Section 3.3.2. We will first present the conceptual model of the framework. Section 4.2 introduces a mechanism to shield the execution schedule from the dynamic events of the real-time system, thereby providing local stability. We have realized that separate bill-of-material and process plans do not provide sufficient information for tight lateral coordination of the supply chain. Section 4.3 addresses how these two data structures can be integrated into a *product network*. Section 4.4 introduces a mechanism for connecting the product networks across agents. Finally, we will discuss two types of inter-agent interaction. The first, which is the type of interaction that occurs between a supply chain entity and entities defined as external to the supply chain, is introduced in Section 4.5. The second, which is the coordination that occurs between agents during operational execution, is introduced in Section 4.6.

4.1 The conceptual model

The conceptual model for our coordination framework is primarily found at the lower coordination level of the MASCOT architecture (introduced in Section 3.3.2). Coordination in the MASCOT architecture is at all levels based on information sharing. However, the two types of information sharing, task sharing and result sharing (defined in Section 3.4.4), both take place within the architecture. High-level coordination is primarily based on task sharing to distribute tasks between organizations. As tasks (e.g., requests for bid) arrive to an entity in the supply chain, the high level agents will negotiate the subtasks with its supplier agents. They might also communicate with the low-level agents through vertical coordination in situations where fine-granularity evaluation (e.g., selective validation of bottleneck decisions) is required as part of the negotiation process. When a satisfactory solution is found, the high-level agents will communicate the result to the low-level agents that are responsible for coordinating the execution of the tasks. This low-level coordination is based on result sharing (through the exchange of constraints). Hence, the high-level agents are responsible for strategic and tactical decision-making while the low-level agents are responsible for operational decision-making. Since this

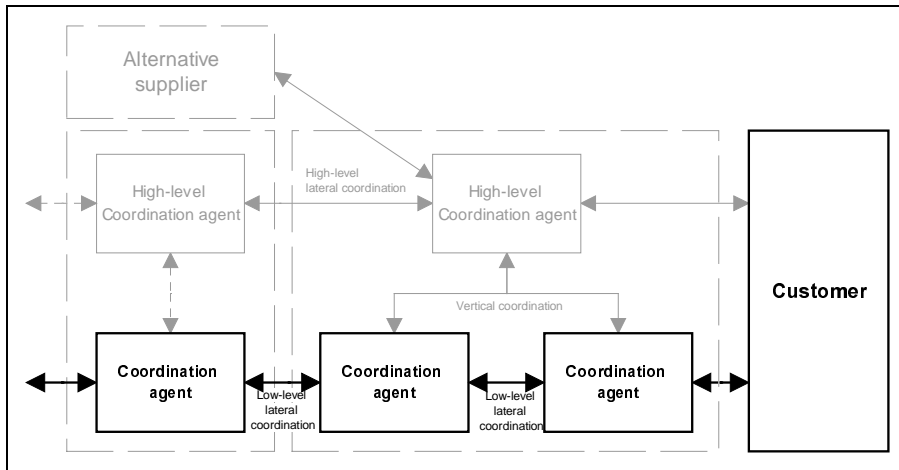


Figure 9: Low-level lateral coordination within the MASCOT architecture.

thesis focuses primarily on the operational aspects of coordination in the supply chain, there will be no further discussion of high-level strategic and tactical issues here. The remaining discussion will thus focus mainly on operational coordination issues. Note, however, that certain decisions which traditionally may have been considered tactical, such as the negotiation of orders, will here be dealt with at the operational level.

4.2 Context mechanism

In the ideal case of an agile production environment, there is no cost (or delay) associated with switching from one schedule to the next. In reality, there are always a number of activities involved each time a schedule is released to the shop-floor, such as producing new priority lists, requesting new tools, replacing the raw material that is waiting in front of machines, or setting up the machines. The release of new schedules should, therefore, be a local decision. In some environments the schedules might be released only once a week, in others once a day, and new schedules might also be replaced in reaction to contingencies that significantly invalidate the currently released context.

This framework supports asynchronous coordination to achieve shop-floor stability and schedule autonomy through the use of multiple *contexts* within each coordination agent. This is illustrated in Figure 10. A context is an independent workspace with its own set of underlying assumptions, called a model. Each agent maintains at any given time (at least) two contexts. The *released context* defines the schedule that currently is released to the shop floor. Real time coordination between

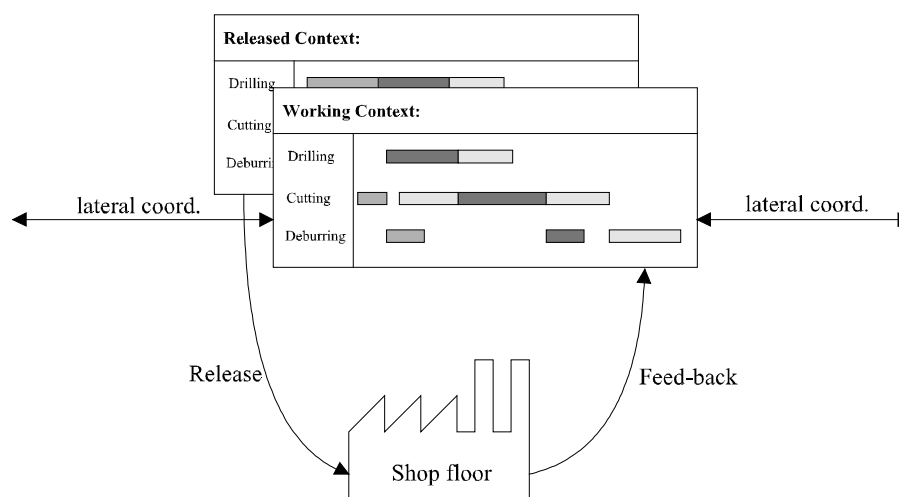


Figure 10: Each agent maintains multiple contexts concurrently. The *released context* directs the execution on the shop-floor while real-time status information from the shop-floor and external agents are collected in the *working context*. The replacing of the released context with the working context is performed asynchronously as a result of a local decision, normally preceded by some form of regeneration of the schedule within the working context.

agents and feedback from the shop-floor are reflected in the *working context*. Releasing a new schedule hence involves replacing the released context with the working context.

4.3 Integration of bill-of-material and process plan

The traditional method used to store *product structures* is the use of a database of bill-of-materials (BOM). A separate database of process plans describes the sequence of actions (and technological requirements) necessary to produce the products. This method dates back to early material requirements planning (MRP) systems, and it is still dominant in today's MRP II and ERP systems. A prerequisite for tight supply chain coordination is the need to know precisely at what step of a plan the material is needed. This requires the BOM and process plans to be integrated into a *product network*, as shown in Figure 11. The idea of product networks is not new insofar as it has already been advocated in OPT (Jacobs 1984).

Our coordination framework relies on such integrated product networks, where the process plans contain information about material requirements and when exactly they are needed. Once a process plan template is realized in an actual order, we talk about the required materials as *supplies* and the product produced as a *demand*. These supplies and demands also have a wider use, as will be described in the next section.

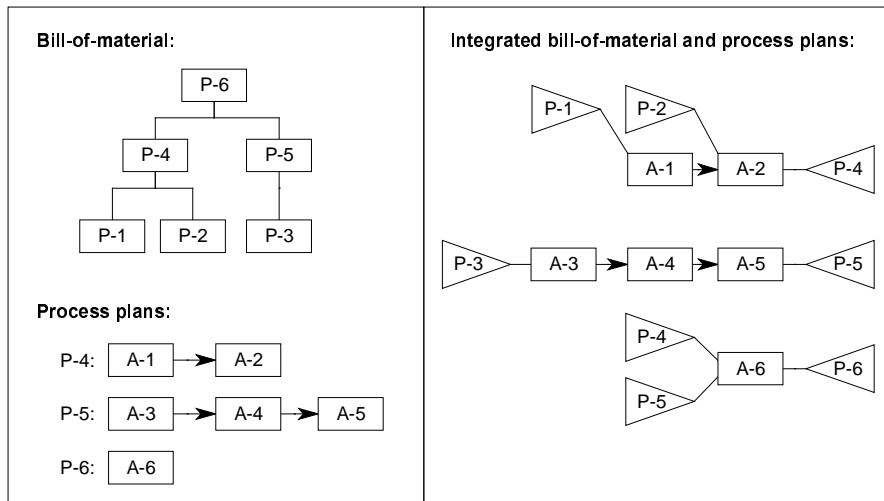


Figure 11: Separated vs. integrated bill-of-material and process plans. Triangles pointing right represent material requirements and triangles pointing left represent the products resulting from the process plans.

4.4 Connectors

Connectors are pairs of corresponding demand and supply objects across agents in the supply chain. The objects are indicated in Figure 12 as triangles. Once created, these objects are assigned unique identifiers that are reported to the other parties. They are therefore referable across agents. When communicating a schedule update, an agent will thus know what agents to address, and the receivers of the information will know which of their local lots the information may affect. The use of such connectors also assists in supporting *lot traceability* all the way back to the source, something which is a federal requirement in certain regulated industries (and is useful for quality management in general).

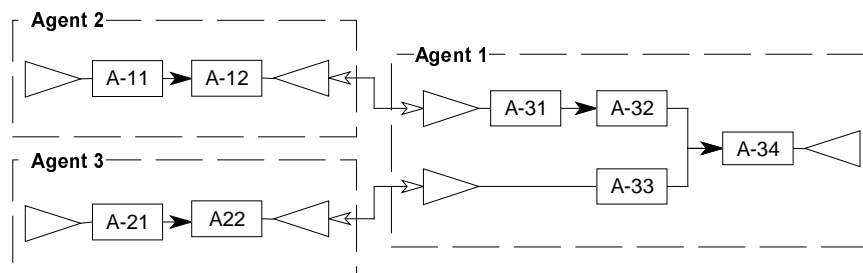


Figure 12: Supply chain coordination connectors. Triangles pointing right represent supplies and triangles pointing left represent demands. Process plans are inserted to fulfill demands from sets of supplies.

A connector creates a one-to-one relation between a demand in the supplier agent and a supply in the consumer agent. It is important to note that even though the connectors define static relations, they do not restrict the supplier or the consumer from *reshuffling* job assignments to the connectors. We have identified situations where a facility can benefit from reshuffling, such as: (1) scrap is identified that will have to be replenished by another job (e.g., a rework order); (2) a job assigned to an urgent demand falls behind schedule (e.g., because of a resource breakdown) and another job (for a less urgent demand) is therefore placed ahead of the first job; or (3) an arriving supply is to be assigned to the most urgent job. In these situations, reshuffling will restore consistency and facilitate the creation of a schedule that can meet the demands. The use of the connectors will also apply to make-to-stock or assemble-to-order environments by allowing for *delayed connection*. A demand object may be temporarily created for a special “forecast” customer in such a situation until the actual order arrives. The demand will then be reassigned to the customer of the newly arrived order.

| Supply object | Demand object |
|--|--|
| <ul style="list-style-type: none"> • Identifier • Identifier for the supplier agent • Identifier of supplier’s corresponding demand object • A list of lots that consume the supply, each entry knowing the node where the lot will require the item • A state variable (e.g., waiting, arrived, or consumed) • The product needed • The quantity needed • The following time points: <ul style="list-style-type: none"> • Promised time — the time reported by the supplier • Release time — the time used internally as a release constraint (not necessarily equal time reported by the supplier) • Scheduled time — when the first of the consuming lots will need the item according to current schedule • Reported time — the time that was last reported/requested to the supplier as a require time | <ul style="list-style-type: none"> • Identifier • Identifier for the customer agent • Identifier of customer’s corresponding supply object • A list of lots that are created to satisfy the demand, each entry knowing the node where the lot is completed • A state variable (e.g., waiting, arrived, or shipped) • The product produced • The quantity produced • The following time points: <ul style="list-style-type: none"> • Required time — the time required by the customer • Due time — the time used internally as a due time (not necessarily the same as the time required by customer) • Scheduled time — when the latest of the supplying lots will complete according to current schedule • Reported time — the time that was last reported/promised to the customer as delivery date • A marginal tardiness cost |

Table 1: Data fields of the supply and demand objects.

4.5 Interaction with external suppliers and customers

Whether there is an established partnership between a supplier and a customer or not, the process of order placement and execution has to be based on some form of contract, and the content of the contract has to be based on some form of negotiation. Standardized communication infrastructures and EDI enable contracting between agents to be more or less automated by the use of contract standards, for example, as defined in ANSI X12, NIIP, or UN/EDIFACT. The scenario in Figure 13 illustrates a typical sequence of interactions for the negotiation of contracts between agents.

The negotiation process starts with the announcement of a request for bid. In established partnerships, the request may be sent to a single or small number of prospective suppliers for the requested parts. Otherwise, the request may be broadcasted, such as by means of electronic marketplaces. The request includes a specification of what the customer wants to purchase, possibly with either an indication of when he wants it delivered or as a long-term blanket order (with some demand rate).

The bidding stage may include some form for technological and economical evaluation of the bid request by the potential suppliers. The evaluation may either result in a rejection of the request, or in a bid being sent to the prospective customer as a response to the request. Depending on the parameters included in the request, the supplier may choose to adjust the bid based on technological constraints or economic considerations.

The awarding stage starts with evaluation of bids. Incoming bids are compared and ranked according to certain criteria. When evaluating bids, the customer may, for example, prefer a bid with a satisfactory delivery date and price from a supplier who over time has proved to be reliable when it comes to meeting his promises. The supplier of the selected bid is granted the order. Detailed circumstances of the orders are documented in a contract that manifests the consensus between the winning supplier and the customer. The bid then turns into an actual order and the losing bidders are informed.

Execution of the order can start once a contract has been established. Bid negotiation is not necessarily ended by order placement. The original contract may allow for some flexibility for both parties to later change the order. The customer may send a formal change order notification to the supplier that the order must be changed in some form (such as delivery date or quantity), or the supplier may send a notification to the customer to inform him about changes due to circumstances introduced after the order placement.

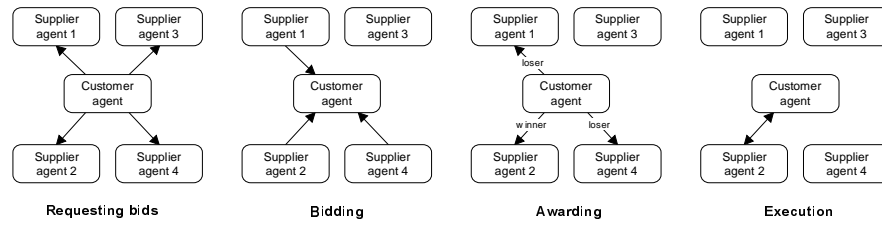


Figure 13: Interaction during bid negotiation. The customer initiates a bidding round by sending requests to a set of potential suppliers. The requests may include time limits for when the bids are due. At that time, the customer may have received bids from a subset of the potential suppliers. After evaluation of the bids, the customer replies by awarding the winning bid and rejecting the others. The resulting contract between the customer and the awarded supplier turns the bid into an actual order subject to execution.

4.6 Coordination

The supply chain coordinates by means of asynchronous exchange of information between its agents. This allows an agent to communicate information directly to other agents. The messages that are received from other agents are parsed into information that is incorporated into the working context as *unresolved issues*. This will thereby be taken into account the next time the schedule is regenerated.

| <i>Performative</i> | <i>Examples</i> |
|---------------------|--|
| <i>ask-if</i> | Requesting a supplier to estimate a delivery date for a potential bid. |
| <i>error</i> | Informing that the received message was not understood. |
| <i>reply</i> | Returning an estimated delivery date for a potential bid. Disavowing a request for bid. |
| <i>tell</i> | Informing a supplier that a bid is approved or canceled. Reporting an updated delivery date to a customer agent. Reporting an updated need date to a supplier agent. |

Table 2: Example of KQML coordination messages.

4.6.1 Asynchronous coordination by means of “unresolved issues”

Real-time scheduling systems are subject to a changing world. What is known as a fact at one point of time might later be invalidated. These systems must, therefore, possess mechanisms for *non-monotonic* reasoning about incoming and sometimes contradictory information. Within our framework, the state of the solution is at any time summarized as a set of *unresolved issues* (Sadeh *et al.* 1998). An unresolved

issue is an indication that a particular aspect of the solution is incomplete, inconsistent, or unsatisfactory.¹³ In light of supply chain coordination, such an issue might be that the promised time from a supplier is inconsistent with the scheduled time of activities needing the supplied material, or that a scheduled order has been canceled. The unresolved issues provide a powerful workflow management mechanism that helps the system (and the user) keep track of inconsistencies in a given context. To update or to refine a particular schedule, a set of unresolved issues is first selected for processing. During processing, new unresolved issues might be generated, for example, canceling an order may introduce an unresolved issue indicating that the schedule now has room for improvements, and regenerating the schedule may introduce an unresolved issue indicating that the new schedule might need to be released.

The processing of unresolved issues may be performed either in a manual mode or in an automated mode of execution depending on the type of unresolved issue and the authority of the source agent. An agent may, for example, provide certain automated services for some other agents to provide quick response, such as checking capacity constraints for a prospective order. The decision of what to automate should, however, remain a local decision.

4.6.2 Coordination by means of high-level agents

The asynchronous coordination mechanism described in the previous section is designed to support fully heterarchical supply chain structures. However, the low-level agents within a single organization represent natural *coalitions* since they share objectives. These coalitions may benefit from coordinating their schedules based on the overall goal of the organization rather than the local goals. This requires a centrally coordinated approach to scheduling. Such centralized approaches fall outside the primary theme of this dissertation. Nevertheless, we have included a brief description of coordinated schedule revision for the sake of better understanding the power of the MASCOT architecture.

The high-level coordination agents of MASCOT (see Figure 8 on page 45) include rough estimates of the capacity and demand of each lower-level agent (e.g., by monitoring key resources or as aggregations of total capacity). In line with the micro-opportunistic search procedure of Micro-Boss, this information can be used to direct scheduling decisions to the most critical low-level agent. Coordinated schedule revision may be obtained, therefore, by asking the agent that appears to be the most

¹³ Our use of unresolved issues is similar to the notion of “flaws” in the agenda-based control architecture demonstrated by Currie and Tate (1991).

critical to perform a single scheduling decision (one cycle in the search procedure), send potential conflicts to the neighboring agents (which stores them in the form of unresolved issues), and report the updated demand profile back to the high-level agent. The high-level agent repeats calling low-level agents until all activities have been scheduled. This procedure enables improvements over the pure asynchronous approach with respect to intra-organizational schedule optimality.

5. Supply chain coordination policies

A supply chain coordination *policy* addresses *when*, *what*, and *with whom* an agent should communicate, and how he can make use of the incoming information. Assumptions regarding the supply chain agents address the level of cooperation that is expected, such as whether the agents are cooperative, self-interested, or hostile, while the environment is characterized by factors such as the level of competition with other supply chains and the level of flexibility in the bid negotiation process. Coordination *mechanisms* defining *how* to communicate have been introduced in Chapter 4.

This chapter proposes a number of different policies applicable to both cooperative and self-interested agents within given environments. Certain of these policies, which we call *synchronization policies*, realize our ideas about how supply chains can be ideally synchronized. Other policies are intended to represent traditional practice in the absence of real time information exchange. These policies will be called *reference policies* since they are used as baselines for comparison with synchronization policies. Each section of the chapter assumes a certain environment. However, the policies defined in a given section differ with respect to the amount of real-time information that is exchanged and thus accessible to each individual agent.

5.1 Just-in-time policies

The first set of policies we will introduce is of a very simple nature. All incoming requests for bid are automatically turned into orders without any modification, that is, a request is never rejected and the submitted bid completely meets the properties requested. The bid will then turn into a contract between the two parties. The resulting production orders are scheduled in a just-in-time manner across the supply chain, that is, without any safety buffers of time or material. These policies establish a natural basis for further refinements in subsequent sections. However, they will be shown to be of conceptual interest, with behavior providing valuable insight into aspects of coordination.

5.1.1 The just-in-time leadtime-based policy (JIT-Lead)

The practice within many companies for estimating prospective delivery-dates is simply to base the estimate on historical leadtime data. Since the sales department

does not have immediate access to (or tools to process) the shop-floor load data, it is restricted to making delivery promises based either on qualitative techniques (relying on judgment, intuition, and subjective evaluation), or on quantitative techniques (e.g., relying on statistical approaches). The JIT-Lead policy relies on a pragmatic quantitative technique based on collected historical data, where the promised delivery date \hat{t}_{comp} is estimated according to

$$\hat{t}_{comp} = \max \left(\max_{i=1}^n (\hat{t}_{comp,i} + \hat{l}_i), t_{req} \right)$$

where i is one of the n parts (sub components or raw materials) needed to produce the requested product, $\hat{t}_{comp,i}$ is the promise date from the supplier of part i , \hat{l}_i is the estimated leadtime for the portion of the process plan where part i is needed, and t_{req} is the requested delivery date. The promised delivery date thus can be easily determined once the leadtimes are estimated. The leadtime may be estimated by considering leadtime data collected for completed *similar* jobs. Significant research in the field of group technology has addressed the issue of similarity between jobs in detail, e.g., Burbidge (1975). However, in our case we will simply assume that two jobs are considered similar if and only if they produce the same part.

Different lot quantities can be expected to entail different leadtimes. The JIT-Lead policy relies, therefore, on a linear regression technique to take lot quantity into account in the leadtime estimates. The linear regression line is calculated as follows: Let $[q_1, l_1]$, $[q_2, l_2]$, ..., $[q_n, l_n]$ be a set of n observations of lot quantities and corresponding leadtimes. Let the average quantity \bar{q} and the average leadtime \bar{l} be given as:

$$\bar{q} = \frac{1}{n} \cdot \sum_{i=1}^n q_i$$

$$\bar{l} = \frac{1}{n} \cdot \sum_{i=1}^n l_i$$

The corrected sum of squares of quantities S_{qq} and the corrected sum of cross products of quantities and leadtimes S_{ql} are given as:

$$S_{qq} = \sum_{i=1}^n (q_i - \bar{q})^2$$

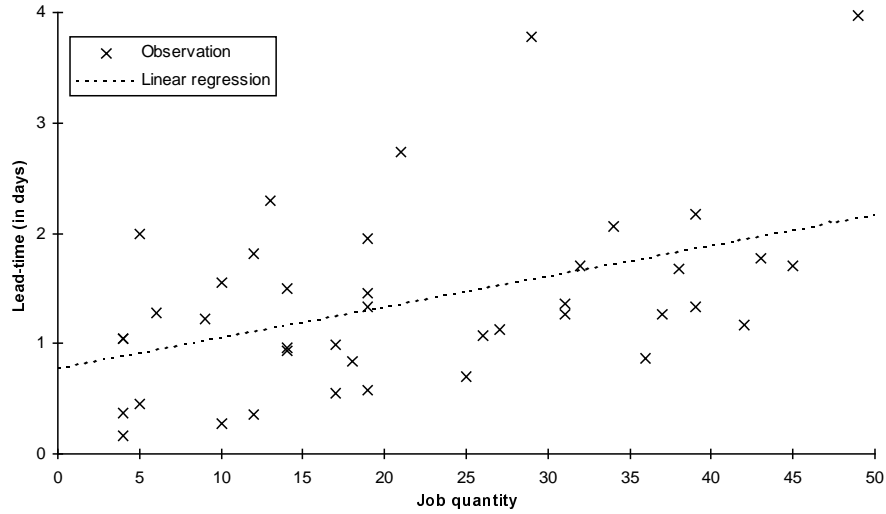


Figure 14: A set of sample quantity-leadtime data for a part type collected during a simulation run.¹⁴ The corresponding linear regression line suggests a positive correlation between quantity and leadtime.

$$S_{ql} = \sum_{i=1}^n l_i \cdot (q_i - \bar{q})$$

The fitted simple linear regression model is then:

$$\hat{l} = \hat{\beta}_0 + \hat{\beta}_1 \cdot q$$

Where estimators for the slope $\hat{\beta}_1$ and the intercept $\hat{\beta}_0$ are given as:

$$\hat{\beta}_1 = \frac{S_{ql}}{S_{qq}}$$

$$\hat{\beta}_0 = \bar{l} - \hat{\beta}_1 \cdot \bar{q}$$

In Figure 14 we find examples where the leadtime for an observation may be 10 times greater than the leadtime for another observation of approximately the same job quantity. Leadtime is thus obviously affected not only by job quantity but also by such other variables as the current situation on the shop floor.¹⁵ The estimation of the

¹⁴ Specific details about the simulation experiments are provided in Chapter 6.

¹⁵ This observation also illustrates why ignoring current shop-floor load when calculating leadtimes (the approach of *infinite* capacity scheduling) will generally fail to provide accurate estimates.

regression line is subject to considerable uncertainty, especially when the underlying data set is small. We accordingly introduce certain additional constraints when constructing the regression line. The intuitive domain knowledge behind these constraints is: (1) The leadtime for producing a part will never be less than zero ($\hat{\beta}_0 \geq 0$), and (2) the leadtime will never decrease as the quantity increases ($\hat{\beta}_1 \geq 0$).

The JIT-Lead policy will make forecasts for the completion of orders according to when they are scheduled to complete.

5.1.2 The just-in-time synchronization policy (JIT-Sync)

The JIT-Sync policy describes the simplest form for real time coordination between supply chain entities. It relies on pure just-in-time coordination. Before an agent regenerates its schedule, it first incorporates new incoming requests for bid (i.e., unresolved issues) into the context. Need dates received from customers become internal due dates, and promise dates received from suppliers become internal release constraints. After the schedule has been regenerated, it communicates scheduled start dates to suppliers (who interprets them as due dates) and scheduled completion dates to customers (who interprets them as release dates).

The JIT-Sync policy will make forecasts for the completion of orders according to when they are scheduled to complete.

5.2 Safety leadtime policies

Safety leadtime is defined in the APICS dictionary (APICS 1987) as “an element of time added to normal leadtime for the purpose of completing a job in advance of its real need date.” In a make-to-order environment, it plays the same role as safety stock in make-to-stock environments. The extra leadtime is inserted into the schedule to absorb unpredictable future events (such as demand surges and machine breakdowns) that interfere with the execution of the schedule without missing the due date. Insertion of safety leadtime will, therefore, increase the robustness of the schedule. The accuracy of forecasts for completion will suffer if the buffer sizes are set inappropriately. The size of time buffers to insert is a tradeoff between the creation of inventory cost (from carrying excessive inventory), tardiness cost (from jobs completed past the due date), and revenue (for coming up with competitive promise dates). With too large buffers the promise dates will either not be competitive, and profitable bids may consequently be lost, or the jobs will be started too early and hence complete before the due date. In the latter case, assuming that finished goods will not be shipped to the customer until the negotiated due date, the

finished goods inventory will thus increase. However, if the buffers are too small, unpredictable events are likely to make a large percentage of jobs tardy. The optimal size of the buffers will balance the marginal increase in profit with the marginal increase in costs.

5.2.1 The time-buffered leadtime-based policy (Buf-Lead)

This policy is based on the JIT-Lead policy, but it allows for safety leadtime to be inserted. Insertion of safety leadtime is obtained by using superfluous leadtimes when estimating the due dates. The Buf-Lead policy uses a statistical approach to determining these leadtimes: If we are able to find the $100(1-\alpha)$ percent confidence interval for the leadtime distribution based on historical leadtime data, we can apply a leadtime for the new bid that contains the confidence interval. Smaller values for α will then translate into higher safety leadtimes. The following procedure for finding the prediction interval for future observations is according to Hines (1990).

We start by assuming that the historical data for leadtimes are normally distributed around the regression line $\hat{l} = \hat{\beta}_0 + \hat{\beta}_1 \cdot q$. Now let

$$S_{ll} = \sum_{i=1}^n (l_i - \bar{l})^2$$

That is, S_{ll} is an estimator of the variance of the distribution of leadtimes. An estimator of the variance σ^2 for $\hat{\beta}_1$ is

$$\hat{\sigma}_{\beta}^2 = \frac{S_{ll} - \hat{\beta}_1 \cdot S_{ql}}{n-2}$$

The $100(1-\alpha)$ percent prediction interval P_{α} for a future observation with quantity q_0 is given as

$$P_{\alpha} = \hat{l} \pm t_{n-2, \alpha/2} \cdot \hat{\sigma}_p$$

where

$$\hat{\sigma}_p = \sqrt{\hat{\sigma}_{\beta}^2 \cdot \left(1 + \frac{1}{n} + \frac{(q_0 - \bar{q})^2}{S_{qq}} \right)}$$

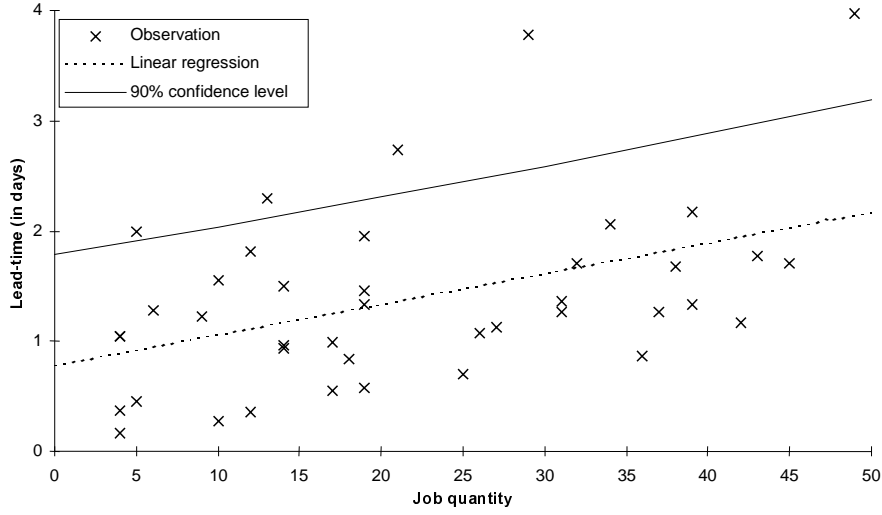


Figure 15: The sample data of Figure 14. The line for 90 percent confidence level may be interpreted as the line for which there is a 90 percent probability that the quantity-leadtime coordinate for a new bid will fall below the line.

Hence, we can insert safety leadtime by estimating the leadtime from

$$\hat{l}_s = \hat{l} + t_{n-2, \beta} \cdot \hat{\sigma}_p$$

where $\beta = 1 - \alpha/2$ is the probability that a future observation will fall below the line given by \hat{l}_s and $t_{n-2, \beta}$ is the β quantile of the t distribution with $n-2$ degrees of freedom. We will call β the *percentile* of the leadtime estimate. Figure 15 shows the same sample data as Figure 14, but it also includes the line for \hat{l}_s with the percentile β set to 0.9. This means that there is a probability of 90 percent that the quantity-leadtime coordinate for the new bid will fall below the line. Note that the slope $\hat{\beta}_1$ of the regression line is an estimator and, consequently, that the distance from the regression line to the line for \hat{l}_s is not constant but widens as we move away from \bar{q} . This means that the farther we extrapolate outside the range of observations, the more uncertain is the estimate. In some cases, the resulting leadtime estimate may become extremely conservative. We have added a rule that identifies these situations. The rule triggers when $\hat{\sigma}_p > \bar{l}$. Since in these rare cases it is assumed that the correlation between quantity and leadtime is too uncertain to rely on linear regression, we estimate the leadtime according to

$$\hat{l}_s = (1 - \beta) \cdot l_{\min} + \beta \cdot l_{\max}$$

where l_{\min} is the shortest leadtime and l_{\max} is the longest leadtime within the set of observations.

The Buf-Lead policy will make forecasts for the completion of orders according to when they are scheduled to complete.

5.2.2 The time-buffered synchronization policy (Buf-Sync)

The Buf-Sync policy is based on the JIT-Sync policy, but it also includes the insertion of safety leadtime. Our approach is to insert safety leadtime into the connectors between agents (see Section 4.4). The amount of safety leadtime to insert depends on the amount of uncertainty to absorb. Safety leadtime is inserted in the following three situations:

- When a supplier promises a completion date, the customer will insert a time buffer of safety leadtime between the promised completion and the scheduled start of the activity.
- When a customer requests a part by a certain date, the supplier will insert a time buffer of safety leadtime between the internal due time and the requested date.
- A forecast for the completion of an order will be made by adding a time buffer of safety leadtime to the scheduled completion of the job.

The first two situations are in accordance with the principle of honesty and faith between agents (discussed in Section 3.4.5). The agent that is the source of the communication sends true information: “This is when I will deliver it according to current schedule,” “This is when I will need it according to current schedule,” and it is up to the receiving agent to make sure it is satisfied: “To be sure the part has arrived when the activity starts, I will wait one day,” “To be sure I can deliver by the requested time, I will try to schedule the part to be ready at least one day before.”

Intuitively, there will be more unpredictable events to handle the farther into the future we look. Appendix B includes a discussion of the amount of safety leadtime to insert under different circumstances. It suggests the use of an “S-shaped” padding function that returns only a small time buffer for values close to current time but converges asymptotically to a maximum time buffer size for large values. This method will be used by the Buf-Sync policy.

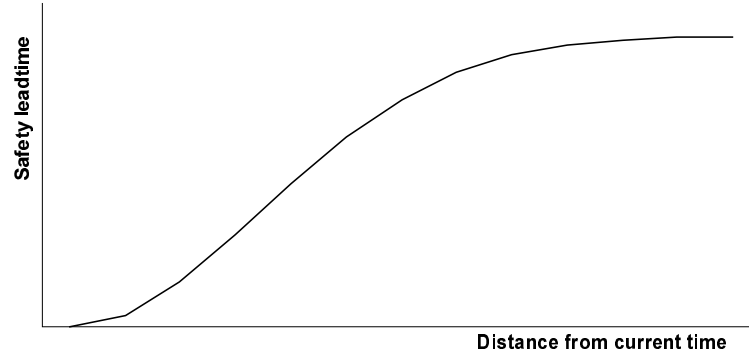


Figure 16: An “S-shaped” function for time buffering.

5.3 Bid refusal policies

Bid refusal policies assume that a manufacturer will only submit a bid if there is reason to believe that the bid contributes to a marginal increase in profit. The policies include a “what-if” check to find the costs associated with the bid. A bid will only be submitted if the revenue associated with the bid is greater than the expected costs, that is, if the sales revenue exceeds the sum of expected holding costs and expected tardiness costs (penalty for completing the order past the date requested). The expected costs \hat{C} for the bid are given as

$$\hat{C} = (\hat{d}_{lead} + \max(0, t_{req} - \hat{t}_{comp})) \cdot mic + \max(0, \hat{t}_{comp} - t_{req}) \cdot mtc$$

where \hat{d}_{lead} is an estimate for the leadtime, \hat{t}_{comp} is an estimate for the completion time, t_{req} is time requested by the customer, mic is the marginal inventory cost, and mtc is the marginal tardiness cost.

5.3.1 The leadtime-based bid refusal policy (Lead-Ref)

This policy is based on the Buf-Lead policy but includes the possibility of refusal by the supplier if there is reason to believe that the bid will not contribute to a marginal increase in profit. Leadtimes and due dates are estimated based on historical data. Forecasts for order completion are made according to when they are scheduled to complete.

5.3.2 The synchronization-based bid refusal policy (Sync-Ref)

This policy is based on the Buf-Sync policy but includes the possibility of refusal by the supplier if there is reason to believe that the bid will not contribute to a marginal increase in profit. Forecasts for order completion are made by adding a time buffer to the scheduled completion of the jobs.

5.3.3 The finite capacity-based bid refusal policy (FCS-Ref)

In some companies the sales department may have access to a finite capacity scheduling system for “what-if” scenarios, such as prediction of the completion date for new requests for bid. A bid request may therefore be checked for capacity constraints in the local (first tier) entity in order to estimate the delivery date. Local release constraints are estimated based on historical leadtime information for suppliers the same way as for the Lead-Ref policy, but the leadtime within the first tier entity is the leadtime found by considering the actual capacity and load. This policy thus represents an intermediate level of sophistication when compared with the Lead-Ref and the Sync-Ref policies. Forecasts for order completion are made according to when they are scheduled to complete.

5.4 Promise date negotiation policies

The policies presented so far assume that due dates requested by the customers are non-negotiable. It might just as well be the case that a customer is open for negotiation when he learns that the bid cannot be completed as requested. It is also better for the supplier’s long-term reputation to provide such information up front rather than to let it come later as a surprise to the customer. We assume that the supplier will never refuse to submit bids even when they are expected to be tardy, but will instead include in the bid a conditional acceptance subject to a revised promise date. If the bid is accepted by the customer, the promise date will become the new due date.

We will first assume that the customers automatically accept all due date changes that result from the bid negotiation process (as if the manufacturer is in a monopoly situation). A version where suppliers are in competition is presented below in Section 5.4.4.

5.4.1 The leadtime-based promise date negotiation policy (Lead-Neg)

This policy is based on assumptions similar to those of the Lead-Ref policy, but instead of sometimes refusing to submit bids, it includes the possibility of promise date negotiation with the customer when there is reason to believe that the requested parts cannot be available until later than what is requested. The revised promise date will always be accepted by the customer.

5.4.2 The finite capacity-based promise date negotiation policy (FCS-Neg)

This policy is based on assumptions similar to those of the FCS-Ref policy, but instead of sometimes refusing to submit bids, it includes the possibility of promise date negotiation with the customer when there is reason to believe that the requested parts cannot be available until later than what is requested. The revised promise date will always be accepted by the customer.

5.4.3 The synchronization-based promise date negotiation policy (Sync-Neg)

This policy is based on assumptions similar to those of the Sync-ref policy, but instead of sometimes refusing to submit bids, it includes the possibility of promise date negotiation with the customer when there is reason to believe that the requested parts cannot be available until later than what is requested. The revised promise date will always be accepted by the customer.

5.4.4 Promise date negotiation with competition

The bid negotiation processes of the Lead-Neg, FCS-Neg, and Sync-Neg policies assume that the customer always accepts the delivery date that the supplier proposed. This may not always be true. In Section 3.1 we discussed the importance of maintaining multiple sources of supply as well as the ability to easily switch between suppliers as two of the five bargaining forces for customers. A customer may be likely to request bids from several alternative suppliers when the negotiation process is similar to the example outlined in Section 4.5.

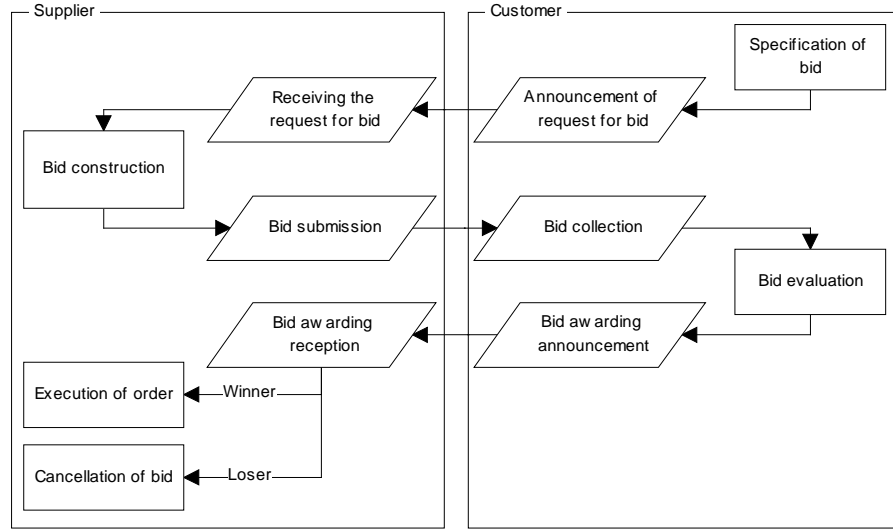


Figure 17: The bid negotiation process for suppliers in competition.

We have created a new scenario that takes this into consideration. The negotiation process is shown in Figure 17. The nature of competition is assumed to be as follows:

- A bid that meets the date requested by the supplier will always be accepted.
- The chance of losing a bid increases as the difference between the revised promise date and the date requested by the supplier increases.
- The higher the price the customer is willing to pay, the stronger the competition is among suppliers. The chance of losing a bid therefore increases with the sales price.

These assumptions are modeled as follows: Let t_{req} be the date requested by the customer, \hat{t}_{comp} be the time promised by the supplier, and $t_{diff} = \hat{t}_{comp} - t_{req}$ be the difference. Furthermore, let p_{diff} be given as

$$p_{diff} = \frac{t_{diff} \cdot c_{bid}}{t_{profitable} \cdot c_{rej}}$$

where c_{bid} is the sales price for the bid, $t_{profitable}$ is the maximum profitable tardiness, and c_{rej} is a constant indicating how much the customer is inclined to reject bids that are promised later than requested. The probability P_{rej} for a bid to be rejected is then given as

$$P_{rej} = \begin{cases} p_{diff} \leq 0: & 0 \\ 0 < p_{diff} < 1: & p_{diff} \\ p_{diff} \geq 1: & 1 \end{cases}$$

The three promise date negotiation policies when under competition are therefore:

- Lead-Neg/C — the Lead-Neg policy under competition
- FCS-Neg/C — the FCS-Neg policy under competition
- Sync-Neg/C — the Sync-Neg policy under competition

6. Empirical evaluation

Chapter 5 proposed a number of different policies applicable within certain environments. Certain of these policies realize our ideas about how supply chains can be ideally synchronized, while other policies establish baselines for comparison. We want to create experiments that make it possible to compare the performance of these policies and enable us to draw conclusions under various conditions.

This chapter is organized as follows: We will first define the scope of the experiments with respect to assumptions made and evaluation criteria used. The supply chain simulation testbed will then be described. Finally, we will present our experiments with the corresponding results. We will first study a two-tier supply chain configuration with one supplier and one customer. Once the validity is proven for this simple configuration, and once we have fine-tuned the coordination parameters, we will apply the policies under different supply chain configurations. The empirical evaluation ends with a study of the sensitivity of the various policies with respect to changes in external conditions.

Due to the large number of experiments generated, we have moved the detailed and complementary information, such as how each of the experiments was set up and the parameter settings, to Appendix A, while certain experimental results of subordinate or indirect importance are found in Appendix B.

6.1 The scope of the experiments

Our experiments are based on the following assumptions:

- The supply chain is assumed to rely on a just-in-time and make-to-order mode of production. There is no stock or buffers of intermediate or finished products and no forecast module to start production ahead of order arrivals. These assumptions may be genuine when the product mix (or level of customization) is high and the demand for each product is highly unpredictable, as is the case for some discrete part manufacturers. See, for example, Bielecki and Kumar's (1988) analysis of the optimality of inventory policies for conditions under which the zero-inventory policy may in fact be optimal.
- Agents in the most upstream tier have an immediate supply of raw material, for example, from outside sources with ample stock. This assumption does not reduce the generality of our experiments. We could simply add another tier of suppliers

ahead of this tier if we wanted to model uncertainty in its supply (a two-tier supply chain where the upstream tier has uncertainty in the supply could experimentally be modeled as a three-tier supply chain configuration).

- Production is subject to unreliability due to resource breakdowns and variations in processing times.
- Orders are never canceled after all parties have accepted them.
- There is no rework, scrap, or rejection of shipped parts, for example, due to quality inspections. However, rework that takes place at the same workcenters where the deficiency was introduced may be modeled as loss of capacity. Such losses of capacity are modeled as resource breakdowns. Separate workcenters for rework (and corresponding delays) are not modeled.
- There is no use of overtime work or subcontracting to catch up with tardy jobs. Although this would exacerbate differences between policies, it is realistic given that overtime and subcontracting do entail overhead costs of their own.
- Resources are always present and available. They are either used for the processing of some activity, idle because there is no job waiting to be processed, or subject to a breakdown (i.e., loss of capacity).
- Process plans within individual entities are linear. Even though our modeling framework supports assembly type of process plans, the current version of the problem generator does not. However, we do model assembly activities with the limitation that all required raw materials must be present before starting the first activity in the process plan, as is the case where *kitting* is used.
- Execution of activities is in accordance with the latest released schedule, that is, the job among waiting jobs selected for execution is the one that is scheduled to take place first. Activities are allowed to start earlier than scheduled if material is available except for the first activity of the most upstream entities, which cannot be started ahead of schedule.
- There is full honesty between the coordination agents, that is, the information that is passed between agents is always correct and no information is withheld or lost. This is based on the assumption that agents believe that the exchange of truthful information will be for their own long-term benefit (see Section 3.4) and that the protocol used for message passing guarantees that the message will arrive to the recipient.
- Transportation leadtimes are assumed to be zero. However, we claim that adding transportation resources and activities to the model (or simply modeling transportation as a time delay) would not significantly change the nature of the experiments. This would be equivalent either to adding an extra tier in the supply chain or to adding extra steps in the process routings.

- Schedules are regenerated each day within all agents. A schedule synchronization controller (to be presented in Section 6.3.3) is used to enable schedule consistency across the supply chain.
- For practical reasons, the requests for bid are collected and processed only once per day, that is, right after all schedules have been regenerated. The coordination policies that check the potential bids for capacity do so by scheduling the bid *on top of* the existing schedule and, hence, without altering what has already been scheduled. A scenario where we allowed bid requests to be managed as they arrive would experimentally behave, therefore, quite similarly.

We are aware that the assumption that “there are no stock or buffers of intermediate or finished products” is quite unorthodox. We have gained some initial insight into how stocks could be maintained by *delayed assignment* of supply to demand. This subject is presented in Section 7.2 as a topic for future research.

6.2 Evaluation of supply chain performance

A successful company is a company able to make money, both in the short and long run. Companies thus have multiple objectives that they try to balance. For the purpose of this study, we define short-term profit as the difference between revenue (through sales) and costs. Long-term revenue is maintained through customer satisfaction, that is, the ability to deliver the right products, in the right quantities, and at the right moment. Costs are kept low by maintaining an effective production system. Thus, the four objectives of “deliver the right products,” “deliver the right quantities,” “deliver at the right moment,” and “minimize costs” are what we call the *corporate objectives*, and overall performance should be evaluated in relation to them. Supply chain performance will affect corporate objectives in several ways. One way to connect supply chain performance to the corporate objectives (based on NEVEM 1989) is shown in Figure 18.

Figure 19 shows another version of a performance model. This model was derived as part of a productivity program in Norway called TOPP (Bredrup 1995). It gives performance as a three-dimensional model where the overall performance is the result of performance along the three dimensions:

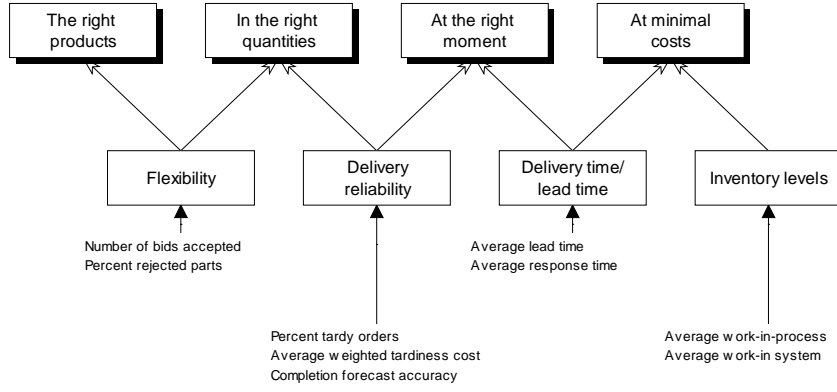


Figure 18: Mapping supply chain performance to corporate objectives.

- Effectiveness — to what extent are customers' (and other stakeholders') needs met?
- Efficiency — how economically are the resources of the company utilized?
- Adaptability — to what extent is the company prepared for future changes?

Even though this performance model is derived for assessing a company's performance, it also makes sense to define supply chain performance in a similar way. In our experiments we evaluate benefits of scheduling and supply chain coordination at the operational level. Variables such as sales price and cost of raw materials are thus considered as given. In addition, we ignore aspects relating to the quality of goods delivered.

We have found that the measurement of the performance of individual entities is difficult in our model since we consider price to be fixed, and since the synchronization policies constantly change the due dates for upstream entities during the execution of contracts. Therefore, the supply chain will be evaluated with respect to overall performance (i.e., social welfare, see Section 3.4.2), the implicit assumption being that rewards are equitably shared between the parties involved.

We will assume that the short and long-term objective of a supply chain is profit maximization. Let S_o be the set of all orders known at time t , $S_c \subseteq S_o$ the set of all completed and shipped orders, R_i the revenue for order i , and C_i the costs for order i . The overall profit P at time t is then defined as:

$$P = \sum_{i \in S_c} R_i - \sum_{i \in S_o} C_i$$

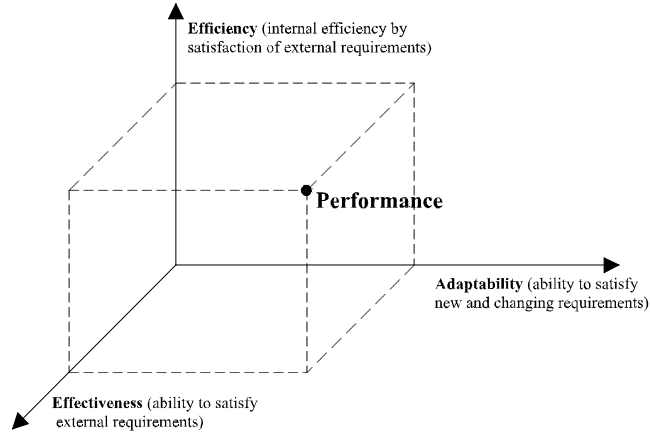


Figure 19: A performance model indicating that the overall performance is the result of performance along three orthogonal dimensions.

Profit is thus given as the difference between sales revenue and costs. Revenue is acquired through deliveries of completed orders to customers. The cost for order i is given as

$$C_i = \sum_{j \in M_i} (C_{i,j}^{fixed} + C_{i,j}^{inventory}) + C_i^{tardiness}$$

where M_i is the set of items required, $C_i^{tardiness}$ is the tardiness cost, $C_{i,j}^{fixed}$ is the fixed cost, and $C_{i,j}^{inventory}$ is the inventory cost for item j . The fixed cost is the cost for acquiring the item. Inventory cost may be due to interest on work in-process and finished goods inventory, the increased risk of obsolescence, etc. Tardiness cost may be due to late delivery penalties, interest on delayed revenue, loss of customer goodwill, etc. Inventory cost and tardiness cost are assumed to be linear. Inventory cost is proportional to the amount of time inventory is carried and tardiness cost is proportional to the amount of time orders are past due, that is:

$$C_{i,j}^{inventory} = mic_j \cdot (\max(t_i^C, t_i^D) - t_{i,j})$$

$$C_i^{tardiness} = mtc_i \cdot \max(0, t_i^C - t_i^D)$$

where mic_j is the marginal contribution to the inventory cost per time unit item j is in the system, t_i^C is the completion time for order i , t_i^D is the due date for order i , $t_{i,j}$ is the time when item j was utilized by order i , and mtc_i is the marginal contribution of tardiness cost per time unit that order i is tardy.

Another performance metric is customer service, which can be measured as the accuracy of forecasts of completion times (or promise dates). This relates to the *pre-expediting* function of a company, that is, the ability to follow up on open orders before the scheduled delivery date in order to ensure timely delivery (of the specified quantity). If the forecast accuracy is high, bid proposals become reliable, orders that are slipping can be identified, and actions can be taken to reduce the severity of the problem (e.g., increasing the capacity of bottleneck resources, or informing the customer about the expected delay). Forecasts of order completion are collected on a daily basis and stored with the order. Once the order is completed, the forecasts can be compared with the actual completion in order to find the corresponding forecast errors. The forecast error $E_{i,k}$ made k days before completion for order i is given as:

$$E_{i,k} = \hat{t}_{i,k}^C - t_i^C$$

where $\hat{t}_{i,k}^C$ is the forecast estimate made k days before completion for order i . The averages and standard deviations for $E_{i,k}$ (grouped by k) can then be obtained at the end of the simulation run.

Profit and forecast errors are the primary metrics for evaluating the policies. We will however also make use of other common performance metrics, such as number of tardy orders, utilization of bottleneck resources, and average leadtime. Table 3 defines the various abbreviations that will be used to report the experimental results.

| <i>Abbreviation</i> | <i>Definition</i> |
|---------------------|---|
| <i>Bids</i> | The total number of requests for bids received. |
| <i>R%</i> | The percentage of bids refused or rejected. |
| <i>Compl</i> | The total number of orders completed. |
| <i>Tardy</i> | The total number of orders completed tardy. |
| <i>U%</i> | Average utilization of the most utilized resource. |
| <i>Lead</i> | Average number of hours from the start of the first activity until the completion of the last activity within the supply chain. |
| <i>Adj</i> | Average due date adjustment made as part of the bid negotiation process. |
| <i>INV</i> | The total in-system inventory cost (including both work-in-process and finished goods inventories). |
| <i>TAR</i> | Total tardiness cost. |
| <i>Revenue</i> | Total revenue from completed orders. |
| <i>Profit</i> | The difference between <i>Revenue</i> and the sum of <i>INV</i> and <i>TAR</i> . |

Table 3: Performance measurement abbreviations used to report experimental results.

6.2.1 Statistical confidence

To draw conclusions from the experimental results, we will perform *all pairwise comparison* of the coordination policies. For any pair (p_a, p_b) of policies, the comparison will establish the level of statistical confidence that p_a will result in a higher profit than p_b . Each coordination policy has been simulated in a series of randomly generated experiments. Because of the randomness of the experiment generator (to be described in Section 6.3.1.1), the experiments will differ with respect to resource requirements. The average profit will, therefore, vary depending on which of the generated experiments we are simulating. The variance reduction techniques (to be described in Section 6.3.1.3) will help identify differences in observed performance for different policies applied to the same experiment due to differences in the policies. For each of the experiments, we have thus calculated x_i as the difference between the profit when using p_a and the profit when using p_b . Therefore, the confidence level for the hypothesis that p_a will result on average in a higher profit than p_b is given as the probability that the mean of the difference distribution is positive. The following is the theory necessary to establish the level of confidence.

Suppose that we have obtained n random observations x_1, \dots, x_n from a statistical distribution with unknown mean μ and unknown variance σ^2 . The sample mean $\hat{\mu}$ and sample variance $\hat{\sigma}^2$ of a statistical distribution can be estimated as

$$\hat{\mu} = \frac{1}{n} \cdot \sum_{i=1}^n x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-1} \cdot \sum_{i=1}^n (x_i - \hat{\mu})^2$$

The sample mean represents a random variable itself. The variance of $\hat{\mu}$ is estimated as

$$Var(\hat{\mu}) = \frac{\hat{\sigma}^2}{n} = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n \cdot (n-1)}$$

Thus, by running a large number of simulations, the variance will approach zero and the sample mean will converge to the real mean. In our case, we will have a relatively small number of observations assuming that x_i is a random variable from the normal distribution $N(\mu, \sigma^2)$ and that t_n is given as

$$t_n = \frac{\hat{\mu} - \mu}{\sqrt{\text{Var}(\hat{\mu})}}$$

The random variable t_n follows a t distribution with $n-1$ degrees of freedom, denoted t_{n-1} . The level of confidence for $\mu < 0$ is now given as

$$P(\mu < 0) = t_{n-1} \left(\frac{\hat{\mu}}{\sqrt{\text{Var}(\hat{\mu})}} \right)$$

The α percent confidence interval, I_α , for μ is the interval that with a probability of α percent contains μ . The interval is given as

$$I_\alpha = \hat{\mu} \pm t_{n-1, 1-\alpha/2} \cdot \sqrt{\text{Var}(\hat{\mu})}$$

where $t_{n-1, 1-\alpha/2}$ is the $1-\alpha/2$ quantile of the t distribution with $n-2$ degrees of freedom

6.3 The simulation testbed

The use of simulation for understanding issues of strategic organizational decision-making has gained considerable attention and momentum in recent years. In Feigin *et al.* (1996), Kumar *et al.* (1993), and Towill *et al.* (1992), simulation is used to evaluate effects of various supply chain strategies on demand amplification. Tzafestas and Kapsiotis (1994) utilize a combined analytical/simulation model to analyze supply chains. Swaminathan (1996) utilizes simulation to study the effect of sharing supplier available-to-promise information.

The approach taken in this dissertation is to use discrete event simulation to compare the performance and appropriateness of different supply chain coordination policies under various conditions. The simulation testbed consists of a *problem generator* and a *simulator*. We have used these two modules to generate a number of problems that aim to be representative of real-world problems and to simulate operations across the supply chain while relying on different policies. Simulation is not the primary focus of this dissertation but is merely used as a mean to evaluate and compare different policies. Accordingly, we limit ourselves in this section to those aspects of the simulation testbed that one would need to be aware of in order to replicate our experiments.

6.3.1 Sources of randomness

We have divided sources of randomness into two categories: randomness in the problem generator and randomness during simulation.

6.3.1.1 Randomness in the problem generator

The problem generator generates stochastic variations of supply chain models. The range of variations is controlled by the parameters listed in Table 4 and the distributions are given in Table 5. The problem generator models the supply chain as a hierarchy of objects, where the top level is the supply chain itself. The supply chain is divided into a given number of *tiers*, and each tier contains a given number of supply chain entities modeled as *agents*. The supply chain also includes a given number of final *products*. The tiers are ordered in such a way that the first tier produces the final products and requires intermediate products (e.g., sub assemblies) from the second tier, the second tier produces intermediate products for the first tier and requires intermediate products from the third tier, etc., until the last tier that is assumed to have instantaneous access to unlimited quantities whenever needed. The parts produced by a given tier are distributed evenly among its internal agents. Each agent includes a given number of *resources* that are required to produce a product, as well as a minimum and maximum number of steps in its process plans. For each resource the minimum and maximum processing time per item are defined. Finally, for each product the minimum and maximum marginal inventory cost per item are given. The grammar for specifying the input model is provided in Table 26 of Appendix A.

| Parameter | Symbol |
|--|---------------------|
| Number of tiers in the supply chain | - |
| Number of agents in a supply chain tier | - |
| Minimum marginal inventory cost per item | mic_{prod}^{\min} |
| Maximum marginal inventory cost per item | mic_{prod}^{\max} |
| Number of resources in an agent | n_{res} |
| Number of different products | n_{prod} |
| Minimum number of activities in a process plan | n_{op}^{\min} |
| Maximum number of activities in a process plan | n_{op}^{\max} |
| Minimum number of raw materials needed by a process plan | n_{BOM}^{\min} |
| Maximum number of raw materials needed by a process plan | n_{BOM}^{\max} |
| Minimum unit duration on a resource | u_{op}^{\min} |
| Maximum unit duration on a resource | u_{op}^{\max} |

Table 4: Parameters for the problem generator.

| Random input variable | Distribution |
|--|--|
| Product assignment to an agent | $p_{agent} = random(\{a_1, \dots, a_n\})$ |
| Number of activities in a process plan | $n_{op} = uniform(n_{op}^{\min}, n_{op}^{\max})$ |
| Resource assigned to an activity ¹⁶ | $r_{op_i} = random(\{r_1, \dots, r_n\} \setminus \{r_{op_1}, \dots, r_{op_{i-1}}\})$ |
| Nominal processing time per part | $u_{op} = uniform(u_{op}^{\min}, u_{op}^{\max})$ |
| Marginal inventory cost per part | $mic_{prod} = uniform(mic_{prod}^{\min}, mic_{prod}^{\max})$ |

Table 5: Summary of probability distributions used by the problem generator. The distribution denoted “random” selects a random element from the set, and “uniform” selects a value from the uniform distribution.

6.3.1.2 Randomness during simulation

The supply chain simulator generates random variations of events during simulation. These events represent stochastic introduction of resource breakdowns, stochastic variations in processing times, and stochastic order arrivals. The range of variations is controlled by the parameters listed in Table 6.

| Parameter | Symbol |
|--|--------------------|
| Average number of incoming requests for bid per period | \bar{n}_{bid} |
| Minimum number of items requested in a bid | q_{bid}^{\min} |
| Maximum number of items requested in a bid | q_{bid}^{\max} |
| Minimum sales price per item | p_{bid}^{\min} |
| Maximum sales price per item | p_{bid}^{\max} |
| Average offset from current time for requested due dates | \bar{t}_R |
| Range for actual duration relative to nominal duration | d_{range} |
| Mean busy time to failure | \bar{t}_{busy} |
| Mean time to repair | \bar{t}_{repair} |

Table 6: Parameters for the simulation testbed.

The arrival of incoming requests for bid is assumed to be a *Poisson process*, that is:

1. Requests for bid arrive one at a time.
2. The number of arrivals for a period is independent of the number of arrivals in previous periods.
3. The distribution for the number of arrivals per period is independent of time.

¹⁶ We will avoid reentrant process plans whenever possible by removing the resource assigned to an activity from the set of resources to select from for subsequent activities.

Since the average number of arrivals per period is assumed to be known, we can apply the Poisson distribution in order to generate the number of arrivals for a period.

The range of variation of incoming requests for bid is due to a number of factors, such as what item is requested, in what quantity it is requested, and by when. We have also assumed that the sales price per item for a part may vary from bid to bid. This may be due, for example, to the use of price differentiation in the sales department. The item requested is assigned randomly from the set of items available. The quantity requested is assumed to be uniformly distributed between a minimum and a maximum quantity. The maximum quantity could, for instance, be the result of historical considerations or economic considerations. For the due date requested by the customer, we have assumed a uniform distribution relative to the arrival time of the requests for bid (i.e., relative to current time, t_c). We are aware that the actual due date distribution might be different in some environments, such as when leadtime estimates are periodically distributed to potential customers. However, it is our experience in simulating various distributions (e.g., exponential distributions) that the behavior of the system depends more on the average value of the requested due date than on the specific shape of the distribution.

For the actual duration of activities we have assumed a *beta* distribution, its mean value being set to the activity's nominal duration.¹⁷ This distribution is commonly assumed, for example, in the PERT method developed by the US Navy in 1958 (Miller 1963). By applying the nominal value as the mean for the distribution, we assume that the nominal value is obtained as, for example, the average of a number of previous runs. It is important to note that the mean does not correspond to the "most likely value" for skew-symmetric distributions. The nominal value should, therefore, never be selected as the duration for "ideal execution" or as the "most likely value" since this would make the schedule unrealistic.

One of the most important sources of uncertainty in many manufacturing environments is associated with machine breakdowns (unscheduled downtime). Random breakdowns result from such events as actual machine failures, parts jams, and broken tools. The percentage of capacity lost due to breakdowns is normally given in the form of mean-time-to-failure (the average *uptime* between two consecutive breakdowns) and mean-time-to-repair (the average time from the onset of a breakdown until the resource is back in operation). In reality, the frequency of breakdowns depends on the utilization of a resource rather than calendar time. Breakdowns are rarely introduced when a resource is idle. By monitoring how much

¹⁷ Other distributions suggested for modeling activity duration are *Gamma* and *Lognormal* (the logarithm of the normal distribution). All these distributions share the common trait of being skew-symmetric, but while the Beta distribution is limited by a maximum value, the Gamma and Lognormal distributions are unlimited.

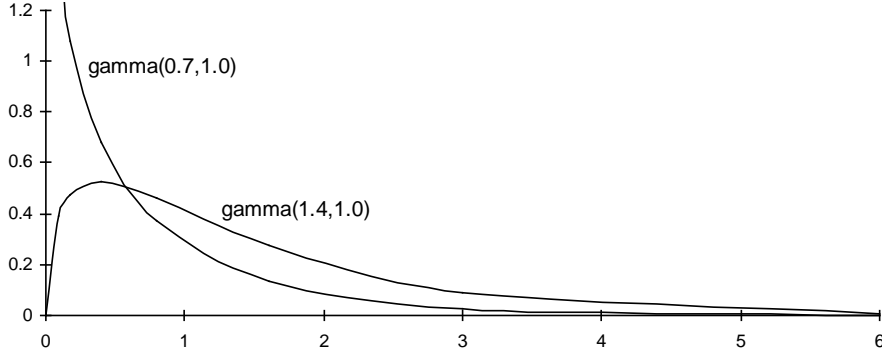


Figure 20: Standard gamma distributions with shape parameters 0.7 and 1.4.

a resource is utilized between two consecutive breakdowns, we can instead obtain data for estimating the distribution of busy-time-to-failure.

The gamma distribution is often suitable as an approximation to both the time-to-failure and time-to-repair functions due to its flexibility (i.e., the fact that its density can assume a wide variety of shapes). Law and Kelton (1992) suggest the use of the gamma distribution with a shape parameter $\alpha_B = 0.7$ in order to approximate busy-time-to-failure and a gamma distribution with a shape parameter $\alpha_R = 1.4$ in order to approximate time-to-repair in their simulation studies of manufacturing systems. We have chosen to model breakdowns according to these guidelines (see Figure 20). The mean of the gamma distribution is the product of the shape parameter α and the scale parameter β . The scale parameter must, therefore, be given as $\beta = \mu/\alpha$ in order to ensure that the gamma distribution has its mean at the intended mean μ .

We have now defined the probability distributions assumed for each of the random input variables. The resulting distributions are summarized in Table 7.

| Random input variable | Distribution |
|---------------------------------|--|
| Requests for bid per period | $n_{bid} = poisson(\bar{n}_{bid})$ |
| Customer's requested due-date | $t_R = t_C + uniform(0, 2 \cdot \bar{t}_R)$ |
| Requested quantity for a bid | $q_{bid} = uniform(q_{bid}^{\min}, q_{bid}^{\max})$ |
| Requested item for a bid | $b_{prod} = random(\{p_1, \dots, p_n\})$ |
| Sales price per item for a bid | $b_{price} = uniform(b_{price}^{\min}, b_{price}^{\max})$ |
| Actual duration for an activity | $d_{op} = u_{op} \cdot q_{bid} \cdot beta(1 - d_{range}, 1 + 2 \cdot d_{range}, 1.5, 3.0)$ |
| Busy time to failure | $t_{busy} = gamma(0.7, \bar{t}_{busy}/0.7)$ |
| Time to repair | $t_{repair} = gamma(1.4, \bar{t}_{repair}/1.4)$ |

Table 7: Summary of probability distributions used to generate random input during simulation. The distribution denoted "random" selects a random element from the set.

6.3.1.3 Variance reduction

If we can somehow reduce the variance of an output random variable of interest without disturbing the expectations, we can obtain greater accuracy, that is, smaller confidence intervals, for the same number of simulation runs. Such techniques to improve the statistical efficiency of simulation are called *variance reduction techniques* (Law and Kelton 1992).

An effective variance reduction technique that applies when we are comparing alternative policies is reduction through *common random numbers*. The basic idea is to simulate the alternatives under similar experimental conditions so that we can be more confident that any observed difference in performance is due to differences between the coordination policies rather than to fluctuations in the experimental conditions. In our experiments, these conditions are the generated random variates used to drive the model through simulated time.

In all cases where we are comparing alternative policies, we have made use of variance reduction through common random numbers in the following way: The random number generator we have implemented is based on a linear congruential algorithm (Sun Microsystems 1993). Consequently, it generates sequences of pseudo-random numbers, where a sequence is predetermined by the initialization entry point (or seed). We have used several independent random number generators, operating simultaneously, each of which is responsible for generating random numbers for a specific class of input random variate. In this way, we have been able to recreate similar experimental conditions for a class of events by re-initializing the corresponding random number generator before the simulation of each policy. The classes of events subject to variance reduction are:

- Arrival of requests for bid — Each policy is subject to identical sets of requests for bid, that is, the same number of requests arriving at the same time, and the requests are identical with respect to requested item, requested quantity, sales price, and requested due-date.
- Introduction of resource breakdowns — Each policy is subject to identical patterns of resource breakdowns, that is, identical sets of busy-time-to-failure and time-to-repair will be generated for each resource.

The sequence in which activities are executed on a resource depends on the coordination policy that is applied. Variance reduction with respect to deviations in activity duration has, therefore, not been enforced.

6.3.2 Estimation of the nominal load

The *efficiency* e of a resource is defined to be the long-run proportion of potential processing time (i.e., parts present and resource not blocked by breakdowns) during which the resource is actually processing parts. It is given by

$$e = \frac{\bar{t}_{busy}}{\bar{t}_{busy} + \bar{t}_{repair}}$$

where \bar{t}_{busy} is the mean-busy-time-to-failure and \bar{t}_{repair} is the mean-time-to-repair. The efficiency is thus a measure of the importance of breakdowns. The number of activities introduced on resource i per bid may be estimated as

$$\hat{n}_i^{op} = \bar{q}_{bid} \cdot f_i^{bid}$$

where f_i^{bid} is the fraction of bids that will introduce an activity on resource i and \bar{q}_{bid} is the average quantity in a request for bid. The nominal load l_i^N is the portion of time when resource i is available for processing that it is expected to be busy:

$$l_i^N = \frac{\bar{n}_{bid} \cdot \bar{u}_{op} \cdot \hat{n}_i^{op}}{d_p \cdot e}$$

where \bar{n}_{bid} is the average number of requests for bid per period, \bar{u}_{op} is the average unit duration on the resource, and d_p is the duration of a period. We will later simulate the different coordination policies subject to variations in nominal load in order to determine the policies' applicability and sensitivity to such variations.

6.3.3 The schedule synchronization controller

The simulation testbed includes a schedule synchronization controller that enables the creation of schedules which are consistent across the entire supply chain. The controller is responsible for triggering schedule regeneration by individual agents in the sequence that is the most effective to achieve consistency between schedules. Whenever called upon, it initiates the regeneration of schedules in a two-pass sequence across the supply chain, first backward then forward. The backward pass ignores release constraints due to material flow over the supply chain. The forward pass is required whenever there are conflicts between the newly generated schedules, and the schedules are again regenerated taking these constraints into account.

The use of a controller is not completely in line with the agent autonomy that we strongly advocated in earlier chapters since individual agents are not allowed to regenerate their schedules asynchronously. In the MASCOT architecture, however, the tasks of the controller could be assigned to the high-level coordination agents, thus being applied within the boundaries of an organization. The use of the controller can also be thought of as representing a policy where every individual agent strives to achieve consistency as soon as it detects inconsistencies with his supplier or customer's schedules.

The reference policies do not exchange synchronization information and, consequently, they only require a single pass to regenerate their schedules.

6.3.4 Initialization

The policies are compared based on their *steady-state* performance. Since a simulation starts with empty schedules in all entities, a *warm-up period* is therefore necessary for the system to move into a steady state, during which no performance statistics are collected. The reference policies rely on historical leadtime information to set due-dates internally within each entity (see Section 5.1.1). This information must include a reasonable number of observations to be reliable. The required duration d_w of the warm-up period can be estimated as

$$d_w \geq \frac{n_H \cdot n_S}{\bar{n}_{bid}}$$

where n_H is the number of historical observations for the same part type required to obtain forecasts of reasonable quality, n_S is the number of different products produced, and \bar{n}_{bid} is the average number of requests for bid per period. If, for instance, there are 20 different products and 4 incoming orders per period and we require 10 leadtime observations, then the warm-up period should be at least 50 periods.

6.3.5 The simulation cycle

The simulation clock is updated using a *fixed-increment-time-advance* mechanism, where each increment is called a simulation cycle. The reason for introducing the cycle is to allow for such events as the arrival of a request for bid and schedule generation to be performed on a regular basis, that is, once per cycle. Hence, a cycle is typically one day long. For each cycle, the following three steps take place:

1. Schedule regeneration — Schedules are created in each agent and coordinated depending on the coordination policy used. The schedule synchronization controller determines the sequence of regeneration. A schedule is regenerated by clearing any previous schedule and rebuilding the schedule according to the Micro-Boss algorithm described in Section 2.3.1.
2. Preparation and negotiation of bids — Once per cycle new requests for bid are prepared into bids that are negotiated with the customers. Bid preparation consists of determining whether or not a bid should be placed and what the content of the bid is. Bid negotiation involves a process resulting in either the customer's acceptance, which creates a contract between the two parties that turns the bid into a confirmed production order, or the rejection of the bid by the customer, which cancels the bid.
3. Simulation of execution — Execution of the schedule for the cycle is simulated in a *next-event-time-advance* fashion. Events that drive the simulation time include: the start of an activity, the completion of an activity, the introduction of a resource breakdown, the resolving of a resource breakdown, material being shipped to customers, and material arriving from suppliers. An activity can start if the following conditions hold: the resource is idle, it is in the queue for the resource, and it has the highest priority among the activities in the queue. Leaf node activities (activities that neither have predecessors nor require material from other activities within the supply chain) are added to the queue when their scheduled start times are reached. All other activities are added to the queue when all their predecessors have completed and all required materials have arrived. The priority of an activity is given by its scheduled start time, that is, the activity with the earliest scheduled start time has the highest priority.

6.4 Results for a simple two-tier model

We have modeled the simple supply chain shown in Figure 21. The supply chain consists of two tiers and each tier contains one entity. Since we are assuming a make-to-order situation, every order negotiated and agreed upon with the external customers will create corresponding production orders in the two entities. We have also made the following assumptions:

- Each entity includes 5 resources with unary (disjunctive) capacity.
- All process plans require all 5 resources in some (random) sequence.
- Each entity includes one bottleneck resource. The nominal load of the bottleneck resources is identical.

Further details are provided in Table 27 of Appendix A.

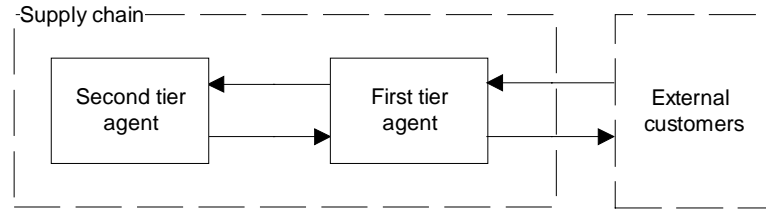


Figure 21: A two-tier supply chain where each tier includes a single entity.

6.4.1 Just-in-time policies

The results reported in this section were obtained by averaging performance over 20 simulation runs for each policy. Table 8 shows performance metrics of the JIT-Lead and JIT-Sync policies. The definitions of the performance metrics are given in Section 6.2. Figure 22 shows the estimated mean values for the profits as circles and indicates the corresponding 95% confidence intervals for the mean profit values as high-low-lines. Figure 23 shows the average forecast errors of the two policies as a function of the number of days prior to completion that the forecasts were made.

We can see that the leadtime-based policy outperforms the synchronization policy for most performance metrics. This is not surprising. Since the leadtime-based policy determine due dates for upstream (supplier) entities based on historical leadtime data, it has the ability to “learn” about the impact of contingencies in the system over time. The synchronization policy does not possess this ability. It tries to schedule jobs just-in-time without any room for contingencies. Every breakdown introduced will, therefore, almost automatically result in tardy orders. This is reflected in a high tardiness cost and consequently reduced profit. However, the synchronization policy outperforms the leadtime-based policy with respect to average leadtime and in-system inventory costs. This is a natural consequence of the synchronization of the supply chain. Since the Micro-Boss scheduling system strives to minimize leadtimes within each entity, and since the schedules of the JIT-Sync policy are feasible across the supply chain, the execution of activities on the shop-floor is likely to be in the sequence they were scheduled, even after breakdowns have been introduced. Jobs are thus executed with minimal leadtime across the supply chain and the work-in-system inventory costs is reduced.

| Policy | Bids | Compl | Tardy | U% | Lead | INV | TAR | Revenue | Profit |
|----------|-------|--------|--------|----------|-------|--------|----------|---------|----------|
| JIT-Lead | 403±8 | 402±8 | 358±18 | 85.9±4.2 | 98±15 | 234±42 | 1632±474 | 3523±85 | 1657±473 |
| JIT-Sync | 403±8 | 391±10 | 356±11 | 85.0±3.0 | 83±5 | 196±16 | 2452±825 | 3426±94 | 778±865 |

Table 8: Performance metrics of the just-in-time policies — two-tier model.

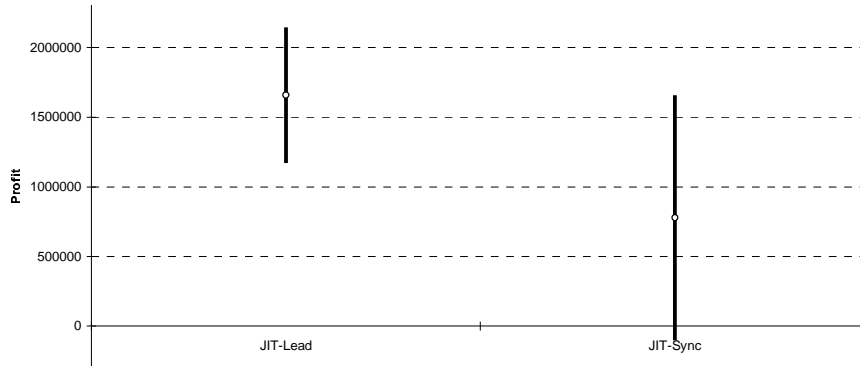


Figure 22: Comparison of average profit for the just-in-time policies.

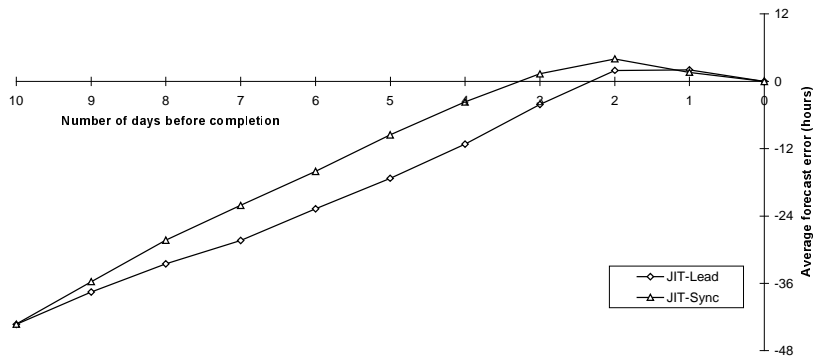


Figure 23: Comparison of average forecast errors for the just-in-time policies. A positive value for the average forecast error indicates that the order on average completed earlier than the forecasted time.

Figure 23 indicates that the two policies result in almost identical patterns of forecast errors, although the JIT-Sync policy tends to forecast orders to complete later (and in general more realistically) than the JIT-Lead policy. Forecasts made more than three days before actual completion tend to be too optimistic in the sense that the estimated completion date is earlier than the actual completion. This type of forecast error appears mainly due to the introduction of new orders between the time when a forecast was made and the time when the order completed. Some of these orders may have high priorities and be scheduled and executed at the expense of lower priority orders that had arrived earlier. The longer the time from a forecast is

made until the order completes and the lower the priority of the order is, the more likely will it be that this can happen. Forecast errors also appear to be due to resource breakdowns and variations in executed duration. However, forecasts made less than three days before completion tend to be cautious, and therefore on the safe side. Orders will on average complete earlier than these forecasts since activities for the 1st tier entity may start ahead of schedule if material is available. Some orders may thus finish before their due-dates, even though they might have been scheduled to complete just-in-time.

6.4.2 Safety leadtime

Before comparing the safety leadtime policies, we sampled the percentile used for the Buf-Lead policy and found 0.7 to be the “optimal” value (see Appendix B). We then compared the performance of the Buf-Lead and Buf-Sync policies on the 20 randomly generated experiments. Table 9 shows the resulting performance metrics. The Buf-Sync policy outperforms the Buf-Lead policy with an average of 33 percent higher profits. This improvement in due date performance is due to the insertion of safety leadtimes. The Buf-Sync policy benefits more from the insertion of safety leadtime than the Buf-Lead policy, the reason being that the Buf-Sync policy is able to apply safety leadtimes selectively while the Buf-Lead policy treats all orders identically independent of the load on the resources.

The majority of the orders are completed past their due-date. Figure 25 shows that the forecasts made by the Buf-Sync policy are on the safe side. The policy is, therefore, aware of potential tardiness when the forecasts are made. However, these forecasts are not utilized to take corrective actions or to negotiate due-dates. The Buf-Sync policy thereby does not reflect the full potential of supply chain coordination.

| Policy | Bids | Compl | Tardy | U% | Lead | INV | TAR | Revenue | Profit |
|----------|-------|-------|--------|----------|-------|--------|----------|---------|----------|
| Buf-Lead | 395±7 | 391±9 | 300±28 | 85.9±4.0 | 99±12 | 238±32 | 1555±477 | 3416±86 | 1623±473 |
| Buf-Sync | 395±7 | 388±8 | 229±31 | 84.2±3.9 | 89±6 | 249±19 | 979±342 | 3400±65 | 2172±325 |

Table 9: Performance metrics of the safety leadtime policies.

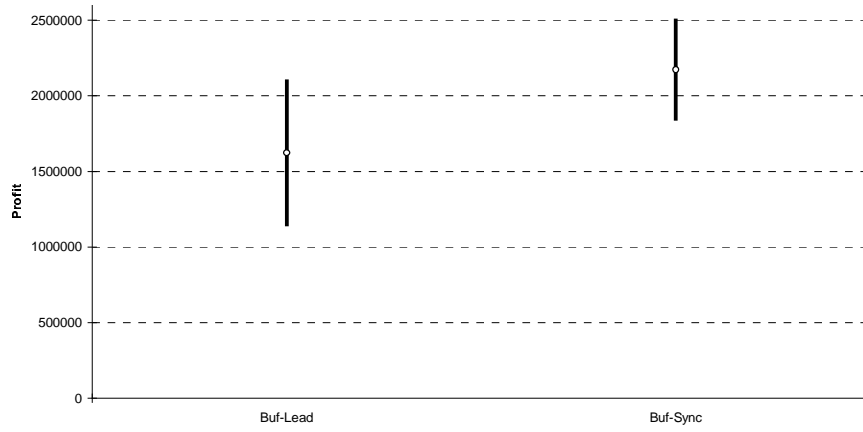


Figure 24: Profit comparison for the safety leadtime policies.

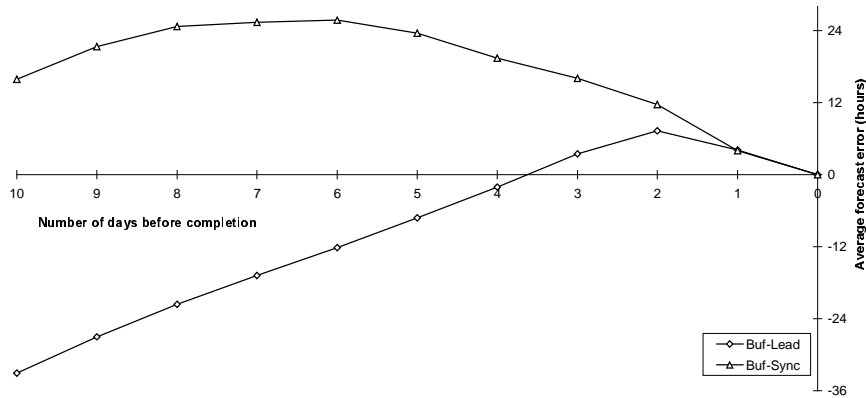


Figure 25: Comparison of average forecast errors for the safety leadtime policies

6.4.3 Refusal of bid requests

We compared the performance of the Lead-Ref, FCS-Ref, and Sync-Ref policies in the 20 randomly generated experiments. Table 10 reports the performance of each policy. Figure 26 shows the average percentage of requests for bid refused by the manufacturer as bars, and the 95% confidence interval for the mean value of profit as high-low-lines.

We can see that even though the Sync-Ref policy rejects more requests for bid than the reference policies, it is able to generate a higher profit. The lower tardiness costs in these experiments outweighs the loss in revenue from the refused bids. We can also see that the use of finite capacity scheduling for decision support during bid preparation makes the FCS-Ref policy more accurate in estimating

completion dates than the Lead-Ref policy (Figure 27). It rejects more requests for bid than the Lead-Ref policy, but compensates for this by completing the accepted orders closer to their due dates. There may, of course, be situations where tardiness penalties are lower than what we have assumed in these experiments. In these situations, the tradeoff between revenue and tardiness costs might favor a policy that rejects fewer requests for bid.

| Policy | Bids | R% | Compl | Tardy | U% | Lead | INV | TAR | Revenue | Profit |
|----------|--------|---------|--------|--------|----------|-------|--------|----------|----------|----------|
| Lead-Ref | 407±11 | 0.9±0.5 | 403±11 | 309±29 | 84.7±4.1 | 95±10 | 234±27 | 1352±381 | 3500±116 | 1913±367 |
| FCS-Ref | 407±11 | 2.9±1.2 | 394±10 | 283±19 | 82.3±3.5 | 86±5 | 209±15 | 866±127 | 3397±101 | 2323±119 |
| Sync-Ref | 407±11 | 8.6±1.6 | 370±9 | 128±8 | 74.7±2.8 | 72±4 | 244±17 | 185±25 | 3101±91 | 2672±89 |

Table 10: Performance metrics of the bid refusal policies.

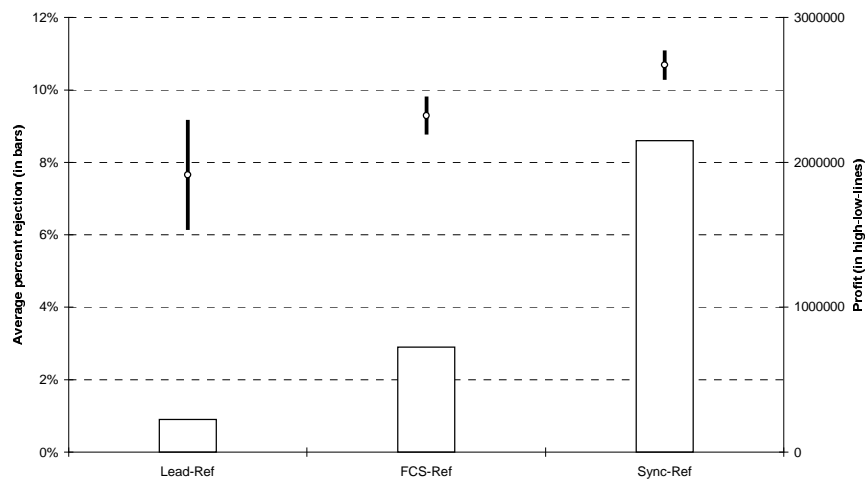


Figure 26: Profit comparison for the bid refusal policies.

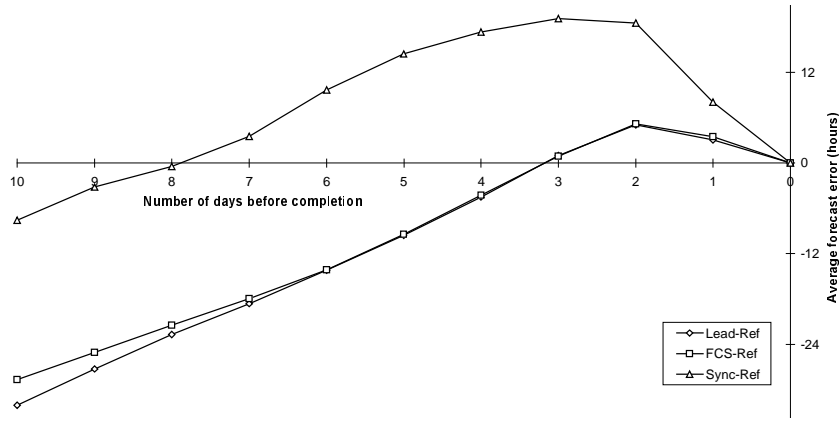


Figure 27: Comparison of average forecast errors for the bid refusal policies.

We have also run a simulation of the 20 experiments that directly compares all seven policies presented so far. These policies share the assumption that the customer's requested due date is non-negotiable. Table 11 and Figure 28 show the resulting performance metrics. The Sync-Ref policy produces on average 16 percent higher profit than the FCS-Ref policy (which is the best of the reference policies). It can also be observed that the size of the 95% confidence interval for the mean value of profit is reduced considerably for the bid refusal policies. The ability to reject bids for which the contribution to the profit is uncertain thus seems to reduce the sensitivity to variations in system input and produce profit more consistently. This ability also has made the percentage of tardy orders drop from 56 percent to 32 percent when comparing the synchronization policies (Buf-Sync vs. Sync-Ref). By rejecting 33 more bids, the supply chain is able to complete 163 more orders on time.

Table 12 gives a pairwise comparison of policies. We can see that the hypothesis that policy Sync-Ref results in a higher average profit than any of the reference policies is confirmed with a 99.9 percent level of confidence.

| Policy | Bids | R% | Compl | Tardy | U% | Lead | INV | TAR | Revenue | Profit |
|----------|-------|---------|-------|--------|----------|-------|--------|----------|---------|----------|
| JIT-Lead | 395±9 | - | 396±8 | 338±18 | 85.7±2.9 | 97±14 | 227±29 | 1422±425 | 3455±82 | 1805±440 |
| JIT-Sync | 395±9 | - | 391±7 | 350±9 | 84.8±2.7 | 83±4 | 196±12 | 2035±535 | 3408±70 | 1176±538 |
| Buf-Lead | 395±9 | - | 394±8 | 297±23 | 84.6±3.2 | 96±14 | 235±27 | 1303±347 | 3442±73 | 1905±357 |
| Buf-Sync | 395±9 | - | 394±9 | 222±28 | 84.8±4.0 | 91±8 | 260±15 | 914±368 | 3446±85 | 2273±380 |
| Lead-Ref | 395±9 | 2.0±1.3 | 393±8 | 279±20 | 84.1±2.9 | 93±8 | 228±18 | 1063±189 | 3410±73 | 2119±191 |
| FCS-Ref | 395±9 | 2.2±0.8 | 386±8 | 270±21 | 83.4±3.0 | 86±5 | 210±15 | 846±105 | 3336±83 | 2280±113 |
| Sync-Ref | 395±9 | 8.3±1.3 | 363±8 | 117±8 | 72.6±2.2 | 68±3 | 237±14 | 164±17 | 3049±70 | 2649±67 |

Table 11: Performance metrics of all policies under the assumption that customer's requested due date is non-negotiable.

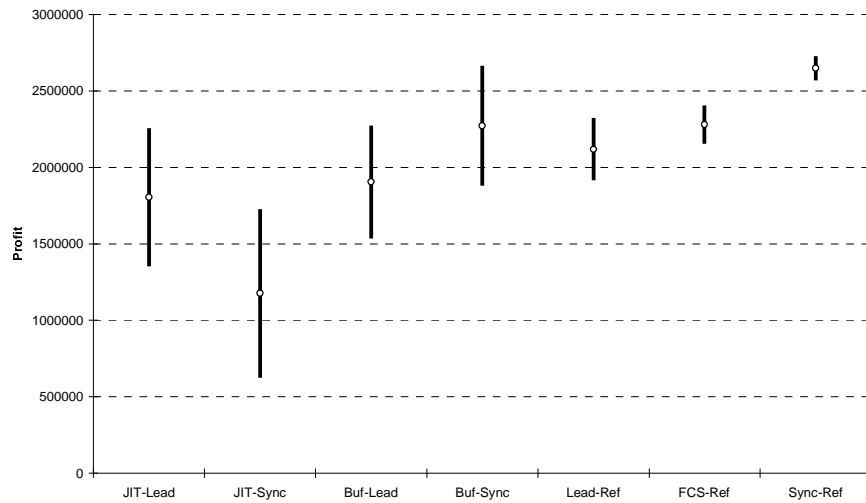


Figure 28: Profit comparison for all policies under the assumption that customer's requested due date is non-negotiable.

| | JIT-Lead | JIT-Sync | Buf-Lead | Buf-Sync | Lead-Ref | FCS-Ref | Sync-Ref |
|----------|----------|----------|----------|----------|----------|---------|----------|
| JIT-Lead | — | 99.89% | 12.87% | 0.00% | 5.74% | 1.36% | 0.03% |
| JIT-Sync | 0.10% | — | 0.00% | 0.00% | 0.02% | 0.01% | 0.00% |
| Buf-Lead | 87.12% | 99.99% | — | 0.04% | 7.49% | 1.38% | 0.01% |
| Buf-Sync | 99.99% | 99.99% | 99.95% | — | 80.14% | 48.29% | 2.24% |
| Lead-Ref | 94.25% | 99.97% | 92.50% | 19.85% | — | 1.20% | 0.00% |
| FCS-Ref | 98.63% | 99.98% | 98.61% | 51.70% | 98.79% | — | 0.00% |
| Sync-Ref | 99.96% | 99.99% | 99.98% | 97.75% | 99.99% | 99.99% | — |

Table 12: Significance of the profit comparison. A table entry gives the probability that the policy in the row results in a higher average profit than the policy in the column.

6.4.4 Promise date negotiation

Before comparing the safety leadtime policies, we sampled the percentile used for the Lead-Neg policy and found a percentile of 0.95 to be “optimal” (see Appendix B). Table 13, Figure 29, Figure 30, and Table 14 show the performance metrics of the Lead-Neg, FCS-Neg, and Sync-Neg policies after simulating execution of the 20 randomly generated experiments. The bars in Figure 29 show the average due date adjustments, that is, the average number of days for the difference between the proposed due date and the requested due date. The high-low-lines show the profit. We can see that the FCS-Neg policy improves upon profit relative to the Lead-Neg policy. The increase in profit is mainly a result of reduced tardiness costs. The

additional use of finite capacity scheduling for decision support during bid preparation makes the FCS-Neg policy propose more realistic due dates than the Lead-Neg policy. However, the Sync-Neg Policy is superior to both of the reference policies concerning profit. Due dates are adjusted more than for the FCS-Neg policy since finite capacity is considered throughout the whole supply chain.

| Policy | Bids | Compl | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|----------|-------|-------|--------|----------|--------|--------|--------|----------|---------|----------|
| Lead-Neg | 396±7 | 394±6 | 277±39 | 84.7±4.5 | 94±8 | 27±5 | 238±29 | 1572±876 | 3443±80 | 1633±869 |
| FCS-Neg | 396±7 | 400±9 | 264±34 | 85.6±4.6 | 104±15 | 83±43 | 254±35 | 887±374 | 3496±92 | 2354±385 |
| Sync-Neg | 396±7 | 392±7 | 203±26 | 83.9±4.1 | 89±9 | 108±44 | 239±21 | 490±238 | 3430±80 | 2700±246 |

Table 13: Performance metrics of the promise date negotiation policies.

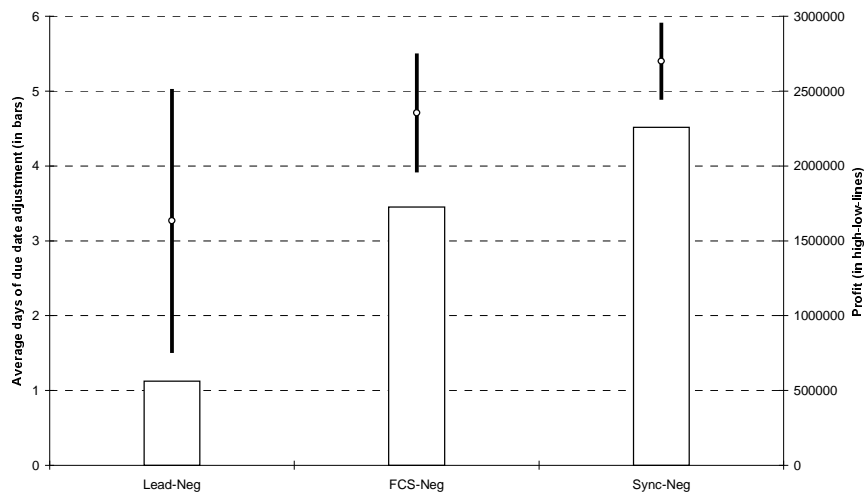


Figure 29: Comparison of average profit and average due date adjustments for the promise date negotiation policies.

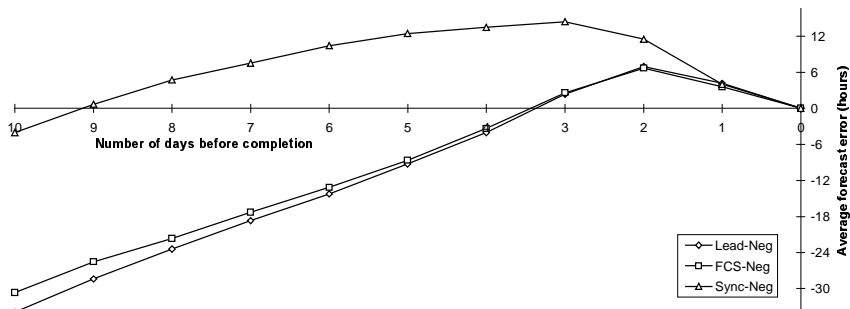


Figure 30: Comparison of average forecast errors for the promise date negotiation policies.

| | Lead-Neg | FCS-Neg | Sync-Neg |
|----------|----------|---------|----------|
| Lead-Neg | — | 0.81% | 0.15% |
| FCS-Neg | 99.18% | — | 0.06% |
| Sync-Neg | 99.84% | 99.93% | — |

Table 14: Significance of profit comparison for the promise date negotiation policies.

6.4.5 Promise date negotiation under competition

Before comparing the Lead-Neg/C, FCS-Neg/C, and Sync-Neg/C policies, we sampled the percentile used for the Lead-Neg/C policy and found a percentile of 0.95 to be optimal (see Appendix B). Table 15, Figure 31, Figure 32, and Table 16 show the results of simulating the policies for promise date negotiation under competition. The bars in Figure 31 indicate the percentage of bids rejected by the customer, and the high-low-lines indicate the profit. The Sync-Neg/C policy rejects more bids than the reference policies. The reason is that the promise dates found by using the Sync-Neg/C policy are feasible with all precedence and capacity constraints in the supply chain. Eventual backlogs of orders are thus taken into account when promise dates are determined. Promise dates made by the Sync-Neg/C policy are on average later than promise dates made by the reference policies since the loads on bottleneck resources are relatively high. As a result, the customer rejects more bids. Still, the average profit for the Sync-Neg/C policy is higher than for the reference policies. From Table 16 we can see that there is a 93 percent probability that the mean profit is higher for the Sync-Neg/C policy than for the FCS-Neg/C policy.

| Policy | Bids | R% | Comp I | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|------------|-------|---------|-----------|--------|----------|------|------|--------|---------|---------|----------|
| Lead-Neg/C | 394±8 | 6.3±1.3 | 374±9 | 198±27 | 78.6±3.5 | 84±7 | 20±3 | 213±18 | 521±212 | 3230±87 | 2496±201 |
| FCS-Neg/C | 394±8 | 6.5±0.9 | 371±8 | 210±25 | 79.3±2.7 | 80±4 | 22±3 | 190±13 | 374±91 | 3188±69 | 2623±100 |
| Sync-Neg/C | 394±8 | 8.9±1.0 | 359±7 | 145±8 | 74.2±2.2 | 68±3 | 29±3 | 188±12 | 154±23 | 3043±69 | 2701±72 |

Table 15: Performance metrics of the promise date negotiation policies under competition.

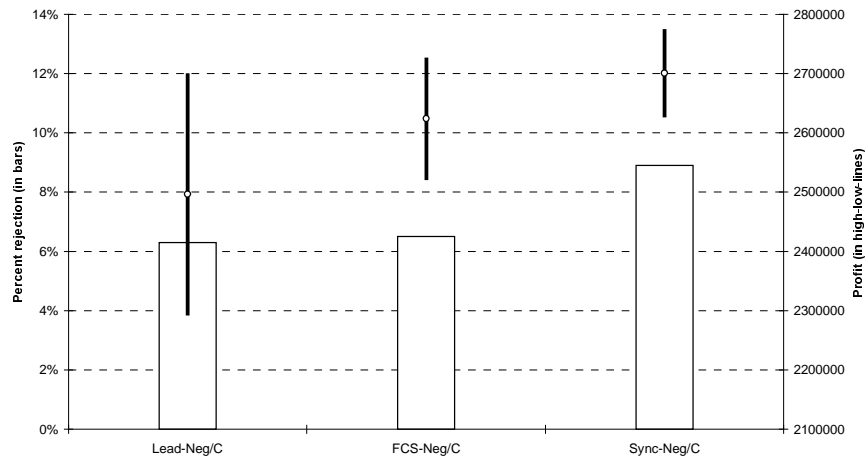


Figure 31: Comparison of average profit and average due date adjustment for the promise date negotiation policies under competition.

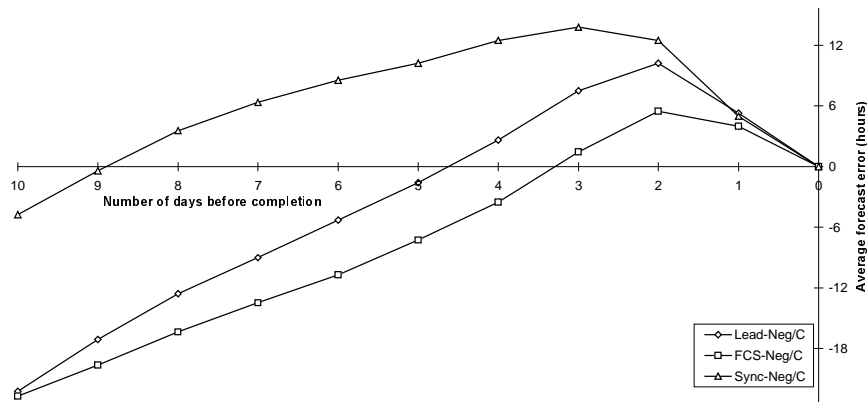


Figure 32: Comparison of average forecast errors for the promise date negotiation policies under competition.

| | Lead-Neg/C | FCS-Neg/C | Sync-Neg/C |
|------------|------------|-----------|------------|
| Lead-Neg/C | — | 6.34% | 2.02% |
| FCS-Neg/C | 93.65% | — | 6.37% |
| Sync-Neg/C | 97.97% | 93.62% | — |

Table 16: Significance of profit comparison for the promise date negotiation policies under competition.

6.5 Results for alternative supply chain configurations

As products become more and more complex and competition increases, one natural consequence is that individual companies will have to focus on the production of core products and purchase more components and sub assemblies from external sources. Consequently, the supply chain network will become more and more complex with respect to the number of tiers and the number of entities in each tier. So far we have explored our supply chain coordination policies as applied to a simple two-tier supply chain configuration with one entity in each tier. We are now ready to examine more complex supply chains. To remain focused, we will limit the discussion and only address policies and assumptions in the context of promise date negotiation under competition.

6.5.1 Long supply chains

The first dimension of configuration change that we will examine involves the introduction of more tiers into the supply chain. Intuitively, in a just-in-time make-to-order supply chain, the more tiers there are ahead of an entity the more the entity will be subject to unreliable supply. However, tight coordination will provide more opportunities to prioritize in the face of delays. We have modeled a five-tier supply chain to represent these longer supply chains, illustrated in Figure 33. We have also made the following assumptions:

- Each supply chain entity includes 5 resources with unary (disjunctive) capacity.
- All process plans require all 5 resources in some (random) sequence.
- The entities in the 1st and 5th tier include one bottleneck resource and the entity in the 3rd tier includes two bottleneck resources. Nominal loads of bottleneck resources are identical.

Further details are provided in Table 28 of Appendix A.

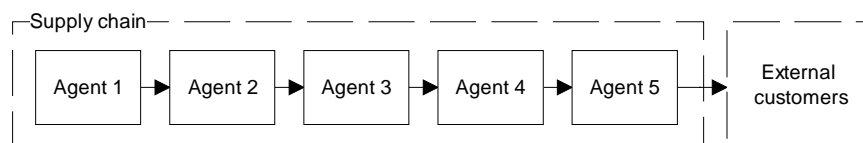


Figure 33: A five-tier supply chain with one entity in each tier.

The results of simulating execution of 20 randomly generated experiments are shown in Table 17 and Figure 34. In this case, the FCS-Neg/C is the policy that rejects the most bids, though the difference in the percentage rejected is relatively small. Moreover, the low bottleneck resource utilization for the Sync-Neg/C policy is striking since the policy does not reject more orders than the reference policies. The most likely reason must be that this policy tends to accept orders that require relatively short processing times to a higher degree than the reference policies and more frequently rejects orders with longer processing times. This might be due to the fact that synchronization policies schedule new requests for bid on top of existing schedules, and that it is easier to fit small activities into the gaps of the schedules. Longer activities, on the other hand, tend to be pushed to the end of the schedule horizon and, therefore, get rejected. Despite this behavior, the Sync-Neg/C policy is able to generate higher profits through significant reductions in both inventory and tardiness costs. Altogether, the benefits of using synchronization policy seem to persist when the supply chain is extended.

| Policy | Bids | R% | Compl | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|------------|-------|----------|-------|--------|----------|--------|------|--------|--------|---------|---------|
| Lead-Neg/C | 388±7 | 12.2±1.4 | 347±7 | 168±20 | 79.9±3.8 | 249±11 | 62±5 | 508±31 | 397±92 | 2991±62 | 2087±85 |
| FCS-Neg/C | 388±7 | 12.7±1.1 | 348±7 | 171±20 | 78.6±3.6 | 240±10 | 62±4 | 489±30 | 381±68 | 2985±63 | 2115±91 |
| Sync-Neg/C | 388±7 | 12.4±0.8 | 341±6 | 152±13 | 74.8±3.5 | 182±6 | 63±4 | 397±20 | 206±33 | 2888±62 | 2286±65 |

Table 17: Performance metrics of different coordination policies when applied to a five-tiered supply chain.

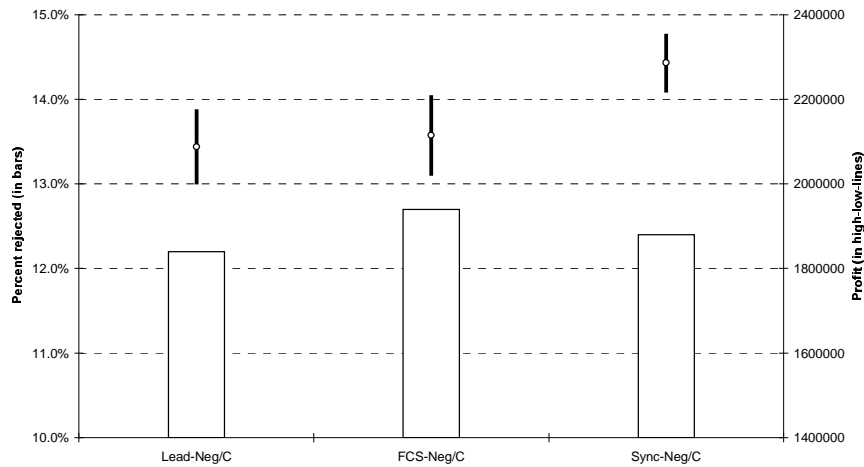


Figure 34: Comparison of rejected bids and profit for different coordination policies when applied to a five-tiered supply chain.

6.5.2 Multiple customers

We will now evaluate policies for supply chain configurations where there is more than one entity in each tier. The first scenario we study involves a supply chain with one supplier and two customers. This configuration is illustrated in Figure 35. Agents 2 and 3 receive requests for bid from external customers. We assume that the entities produce non-overlapping sets of products and, consequently, that a request will be given to either Agent 2 or Agent 3, and not to both of them at the same time. Therefore, they do not compete for bids, and they have independent objectives. Improved performance for Agent 2 is of no benefit for Agent 3, and vice versa. Furthermore, we have made the following assumptions:

- Each entity includes 5 resources with unary (disjunctive) capacity.
- All process plans require all 5 resources in some (random) sequence.
- Each entity includes one bottleneck resource, and the nominal load of the bottleneck resources is approximately identical.

Further details are provided in Table 29 of Appendix A.

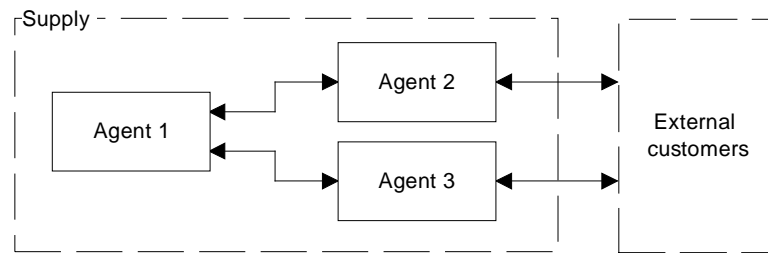


Figure 35: A two-tier supply chain with two entities in the 1st tier and one entity in the 2nd tier.

The results of simulating execution of 20 randomly generated experiments are shown in Table 18, Figure 36, and Table 19. Surprisingly, profits obtained with the FCS-Neg/C policy are almost identical to those of the Sync-Neg/C policy. Since the FCS-Neg/C policy rejects fewer orders, it is able to generate more revenue at the sacrifice of an increase in costs. Furthermore, this specific configuration results in a relatively stable load on the second-tier entity compared to that on the first tier entities. Every request for bid that results in an order will have a corresponding production order in Agent 1, while the orders are randomly distributed between Agents 2 and 3. This stable load of Agent 1 makes the history-based leadtime estimates fairly accurate, and with limited room for improvement through checking

for capacity during bid construction. The Lead-Neg/C policy will, however, suffer from not checking for capacity within Agents 2 and 3, where the fluctuations in load are higher.

| Policy | Bids | R% | Compl | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|------------|-------|----------|--------|--------|----------|--------|------|--------|----------|----------|----------|
| Lead-Neg/C | 399±8 | 12.8±4.2 | 358±16 | 211±23 | 84.8±5.1 | 151±41 | 28±3 | 332±58 | 1147±568 | 3077±138 | 1599±697 |
| FCS-Neg/C | 399±8 | 10.8±2.2 | 359±12 | 200±20 | 83.8±5.1 | 109±8 | 34±5 | 248±20 | 379±106 | 3038±115 | 2412±140 |
| Sync-Neg/C | 399±8 | 13.4±2.2 | 345±11 | 141±9 | 81.1±4.8 | 89±3 | 42±4 | 228±14 | 212±41 | 2884±107 | 2444±125 |

Table 18: Performance metrics.

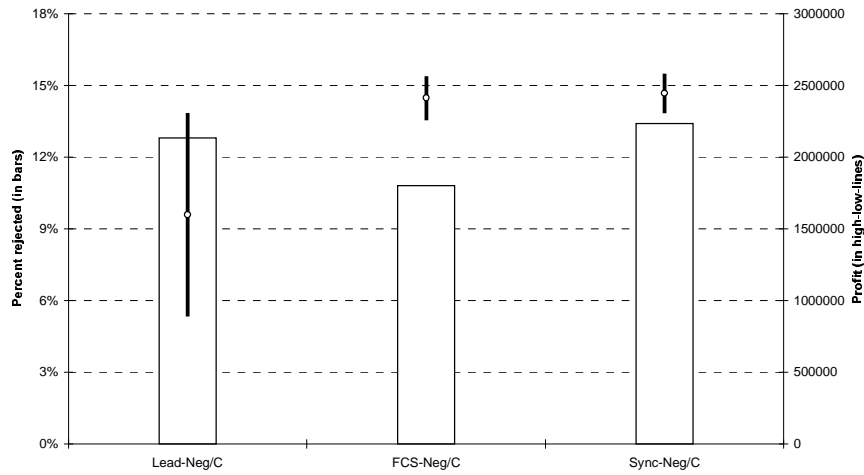


Figure 36: Comparison of rejected bids and profit.

| | Lead-Neg/C | FCS-Neg/C | Sync-Neg/C |
|------------|------------|-----------|------------|
| Lead-Neg/C | — | 0.43% | 0.47% |
| FCS-Neg/C | 99.56% | — | 23.82% |
| Sync-Neg/C | 99.52% | 76.17% | — |

Table 19: Significance of profit comparison.

6.5.3 Multiple suppliers

We will now study the behavior of a supply chain where a customer has more than one supplier. The configuration in Figure 37 is modeled to exemplify this type of situation. The model includes the following assumptions:

- Entities 1 and 2 produce non-overlapping sets of products and, therefore, do not compete for orders.
- Each entity includes 5 resources with unary (disjunctive) capacity.
- All process plans require all 5 resources in some (random) sequence.
- Each entity includes one bottleneck resource, and the nominal load of the bottleneck resources is approximately identical.

Further details are provided in Table 30 of Appendix A. An added potential benefit of synchronizing this kind of supply chain lies in enabling disturbances to be reflected across the supplier segment. The arrival of a high priority order for which Agent 1 is the supplier might, for instance, result in changes in Agent 3's schedule. Synchronizing schedules can make these changes visible for Agent 2 so that he can re-prioritize accordingly.

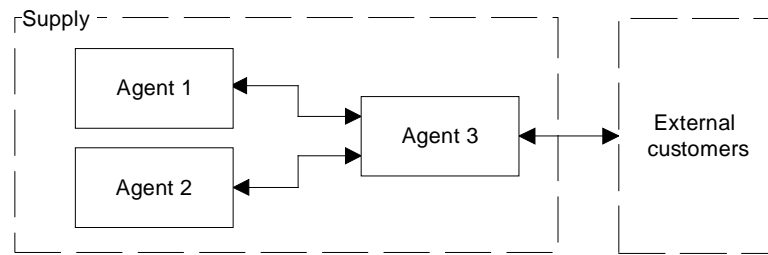


Figure 37: A two-tier supply chain with one entity in the first tier and two entities in the 2nd tier.

The results of simulating execution of the 20 randomly generated experiments are reported in Table 20, Figure 38, and Table 21. The synchronization policy now performs significantly better than both reference policies. We can reverse the argumentation in Section 6.5.2 to explain this behavior: The synchronization policy benefits from checking its bids for capacity on Agents 1 and 2, which have higher fluctuations in load than Agent 3. It is able, therefore, to significantly reduce costs due to both shorter leadtimes and less tardiness.

| Policy | Bids | R% | Compl | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|------------|-------|----------|--------|--------|----------|-------|------|--------|----------|---------|----------|
| Lead-Neg/C | 403±6 | 8.5±1.3 | 369±9 | 229±21 | 85.6±4.4 | 111±7 | 28±3 | 257±25 | 1364±880 | 3215±77 | 1593±922 |
| FCS-Neg/C | 403±6 | 11.3±2.6 | 360±10 | 232±12 | 83.9±5.4 | 110±6 | 36±7 | 243±23 | 847±351 | 3122±95 | 2032±425 |
| Sync-Neg/C | 403±6 | 12.6±1.8 | 353±8 | 145±8 | 77.8±4.3 | 89±3 | 41±4 | 228±18 | 180±27 | 3013±80 | 2605±93 |

Table 20: Performance metrics.

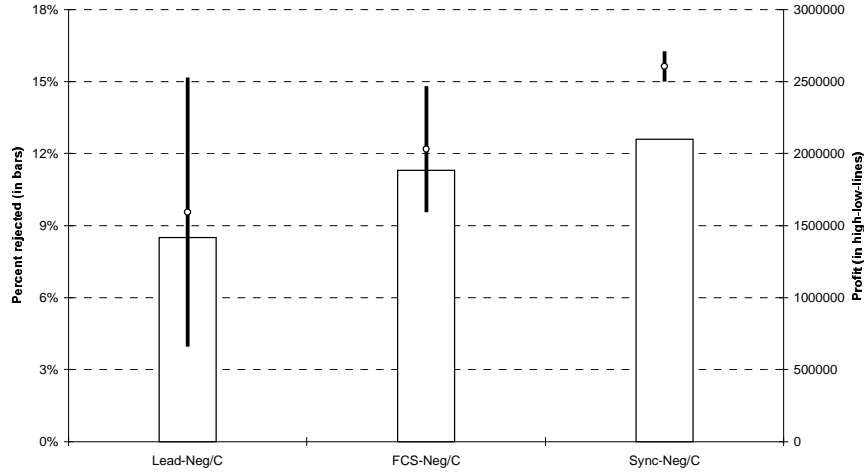


Figure 38: Comparison of rejected bids and profit.

| | Lead-Neg/C | FCS-Neg/C | Sync-Neg/C |
|------------|------------|-----------|------------|
| Lead-Neg/C | — | 5.53% | 1.13% |
| FCS-Neg/C | 94.46% | — | 0.16% |
| Sync-Neg/C | 98.86% | 99.83% | — |

Table 21: Significance of profit comparison.

6.5.4 A larger example

We have crated a model for a larger supply chain as shown in Figure 39. The purpose of the model is to represent a coordination problem that is closer to real world situations. The model is scaled up along several dimensions and includes all the aspects of supply chain configurations presented in earlier parts of this chapter. The supply chain is long enough to capture problems of longer supply chains, and

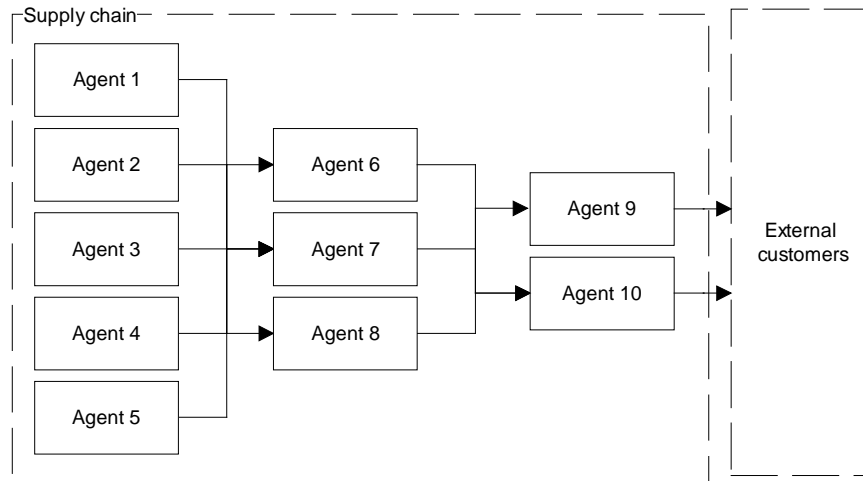


Figure 39: A larger supply chain configuration.

individual entities in the supply chain may have multiple customers and multiple suppliers. The model also includes two aspects not discussed previously:

- The supply chain includes assembly types of process plans. The agents in the 1st and 2nd tier (Agent 6-10) include process plans requiring either one or two types of raw materials. For each order in the 1st tier there will thus be an average of 1.5 orders in the 2nd tier and 2.25 orders in the 3rd tier. For assembly type orders there is an additional challenge in coordination between suppliers. Appropriate coordination may enable a supplier to prioritize his activities based on status considerations for other suppliers. For example, when an entity responsible for assembly processes is informed that a supplier is expected to be late with his deliveries, this information can be sent to the other suppliers for the same order so that they can prioritize accordingly. We will assume that the suppliers are not competing in the same market since they are producing different products and, consequently, that improved coordination mutually benefits all parties involved.
- We allow heterogeneous properties for entities within a tier. Agents differ with respect to number of resources, number of steps in the process plans, and bottleneck severity. Therefore, some of the entities in a tier may be significantly more loaded than others, and some of the entities may be more reliable in their deliveries than others.

Further details about the model are provided in Table 31 of Appendix A.

The results of simulating execution of 20 randomly generated experiments are reported in Table 22 and Figure 40. The performance metrics are again in favor of the synchronization policy. They also seem to indicate that the benefits of synchronizing the supply chain will even increase with the size and complexity of the supply chain.

| Policy | Bids | R% | Compl | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|------------|--------|----------|--------|--------|----------|--------|------|----------|-----------|----------|-----------|
| Lead-Neg/C | 806±10 | 11.5±1.4 | 705±14 | 507±30 | 92.3±2.7 | 209±29 | 36±4 | 1697±230 | 3985±1378 | 6075±147 | 393±1678 |
| FCS-Neg/C | 806±10 | 12.4±1.9 | 699±15 | 503±34 | 90.9±3.5 | 197±24 | 37±4 | 1595±207 | 3386±1118 | 6022±155 | 1041±1387 |
| Sync-Neg/C | 806±10 | 15.2±1.2 | 689±12 | 470±21 | 86.4±3.4 | 147±9 | 45±4 | 1158±94 | 1154±189 | 5765±130 | 3454±346 |

Table 22: Performance metrics.

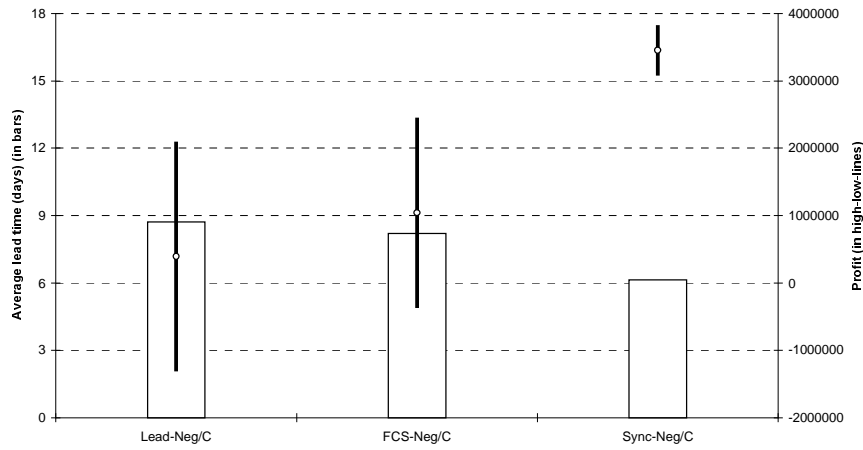


Figure 40: Profit comparison.

6.6 Results for variations of external conditions

6.6.1 Variations in nominal load

We have tested the larger supply chain from Section 6.5.4 for variations in nominal load. The results are shown in Figure 41, Table 23 and Table 24. The nominal load l_N reported is the average load of the most significant bottleneck resource in the supply chain if all requests for bid result in actual orders. The different loads have been sampled by varying \bar{n}_b , the average number of requests for bid per period. We can make the following conclusions from the experiment:

- The policies result in approximately the same supply chain performance when the nominal load is low (i.e., less than 0.6). This may be explained in the following way: Most jobs can be scheduled just-in-time without creating resource conflicts when the nominal load is low. The benefits from checking for capacity as part of the bid negotiation process are, therefore, limited. Furthermore, the average queue size of parts that are waiting to be processed by a resource depends heavily on the nominal load, which means that improved coordination is of limited help for prioritization during the execution of the schedules.
- The synchronization policy is able to consistently maintain a high profit as the nominal load increases above 0.6. Theoretically the profit should increase as the load increases since the supply chain can then pick and choose between a larger number of bid requests, selecting only the most profitable ones. However, the policy places bids as a response to every request without considering the possibility that future and more profitable bids later may arrive. Adding such considerations (i.e., an enhanced version of the refusal process of the Sync-Ref policy) could probably be beneficial in such situations.
- The performance of both reference policies deteriorates as the nominal load increases above 0.6. The reference policies reject close to the same percentage of bids as the synchronization policy. This surprising strength of degradation can only be explained as a result of poor coordination. Without checking new bids for capacity, there is no means to identify the high backlog of orders that is carried and, consequently, the estimates for completion will become unrealistic.

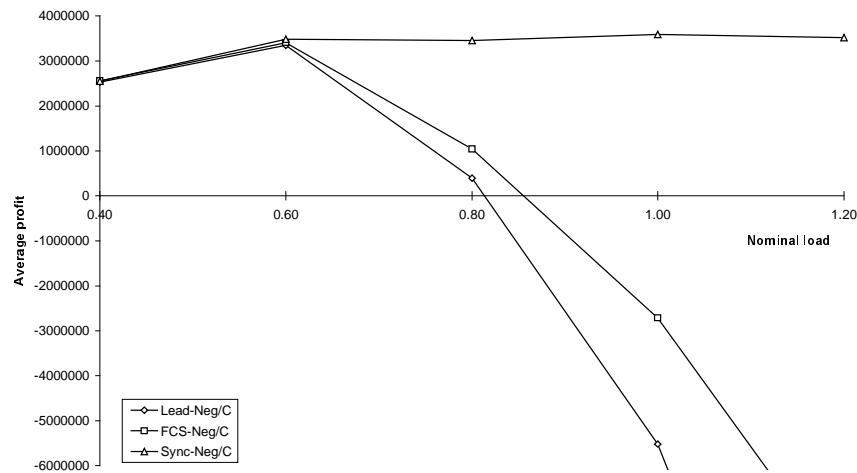


Figure 41: Sensitivity to variations in number of bid requests.

| Policy | I_N | Bids | R% | Compl | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|------------|-------|-------------|--------------|------------|------------|--------------|------------|----------|--------------|----------------|--------------|-----------------|
| Lead-Neg/C | 0.40 | 398 ±10 | 4.9 ±0.6 | 377 ±11 | 155 ±18 | 49.5 ±2.9 | 103 ±5 | 19 ±1 | 535 ±51 | 217 ±46 | 3281 ±124 | 2530 ±96 |
| | 0.60 | 605 ±10 | 6.9 ±0.7 | 564 ±10 | 304 ±17 | 73.1 ±4.3 | 128 ±7 | 25 ±3 | 911 ±90 | 617 ±130 | 4872 ±93 | 3345 ±207 |
| | 0.80 | 806 ±10 | 11.5 ±1.4 | 705 ±14 | 507 ±30 | 92.3 ±2.7 | 209 ±29 | 36 ±4 | 1697 ±230 | 3985 ±1378 | 6075 ±147 | 393 ±1678 |
| | 1.00 | 993 ±18 | 16.6 ±3.0 | 790 ±28 | 624 ±53 | 95.5 ±2.7 | 291 ±38 | 45 ±5 | 2472 ±338 | 9732 ±2916 | 6685 ±230 | -5519 ±3293 |
| | 1.20 | 1207 ±17 | 27.1 ±4.4 | 824 ±30 | 709 ±40 | 96.2 ±1.9 | 388 ±31 | 53 ±6 | 3306 ±325 | 19654 ±4642 | 6988 ±268 | -15971 ±4891 |
| FCS-Neg/C | 0.40 | 398 ±10 | 5.0 ±0.5 | 379 ±12 | 166 ±12 | 49.8 ±3.3 | 103 ±5 | 18 ±1 | 524 ±47 | 221 ±31 | 3297 ±129 | 2552 ±102 |
| | 0.60 | 605 ±10 | 7.1 ±0.8 | 566 ±8 | 308 ±22 | 73.7 ±4.7 | 126 ±7 | 24 ±2 | 900 ±95 | 604 ±138 | 4908 ±72 | 3405 ±201 |
| | 0.80 | 806 ±10 | 12.4 ±1.9 | 699 ±15 | 503 ±34 | 90.9 ±3.5 | 197 ±24 | 37 ±4 | 1595 ±207 | 3386 ±1118 | 6022 ±155 | 1041 ±1387 |
| | 1.00 | 993 ±18 | 19.7 ±2.6 | 782 ±32 | 627 ±44 | 95.5 ±1.8 | 269 ±31 | 53 ±4 | 2239 ±278 | 7044 ±1803 | 6573 ±256 | -2710 ±2205 |
| | 1.20 | 1207 ±17 | 32.1 ±3.6 | 804 ±33 | 653 ±47 | 95.7 ±2.7 | 350 ±27 | 71 ±4 | 2888 ±306 | 12281 ±2357 | 6727 ±314 | -8442 ±2597 |
| Sync-Neg/C | 0.40 | 398 ±10 | 4.7 ±0.5 | 379 ±11 | 203 ±12 | 49.7 ±3.2 | 98 ±5 | 17 ±1 | 510 ±47 | 230 ±28 | 3285 ±115 | 2545 ±102 |
| | 0.60 | 605 ±10 | 8.1 ±0.9 | 558 ±8 | 330 ±16 | 71.0 ±3.9 | 117 ±7 | 28 ±2 | 827 ±77 | 464 ±67 | 4773 ±86 | 3482 ±156 |
| | 0.80 | 806 ±10 | 15.2 ±1.2 | 689 ±12 | 470 ±21 | 86.4 ±3.4 | 147 ±9 | 45 ±4 | 1158 ±94 | 1154 ±189 | 5765 ±130 | 3454 ±346 |
| | 1.00 | 993 ±18 | 22.5 ±1.8 | 767 ±23 | 448 ±26 | 89.5 ±2.4 | 166 ±11 | 59 ±4 | 1390 ±117 | 1190 ±395 | 6170 ±186 | 3590 ±550 |
| | 1.20 | 1207 ±17 | 30.5 ±2.2 | 835 ±24 | 461 ±25 | 93.6 ±1.5 | 192 ±17 | 71 ±3 | 1606 ±136 | 1477 ±706 | 6600 ±222 | 3517 ±904 |

Table 23: Performance metrics of variations of nominal load.

| | Lead-Neg/C | | | | | FCS-Neg/C | | | | | Sync-Neg/C | | | | |
|--------------|------------|-------|-------|-------|-------|-----------|-------|-------|-------|-------|------------|------|------|------|------|
| Nominal load | 0.40 | 0.60 | 0.80 | 1.00 | 1.20 | 0.40 | 0.60 | 0.80 | 1.00 | 1.20 | 0.40 | 0.60 | 0.80 | 1.00 | 1.20 |
| Lead-Neg/C | — | — | — | — | — | 11.02 | 11.80 | 4.72 | 0.17 | 0.00 | 25.22 | 0.78 | 0.01 | 0.00 | 0.00 |
| FCS-Neg/C | 88.97 | 88.19 | 95.27 | 99.82 | 99.99 | — | — | — | — | — | 67.23 | 5.35 | 0.02 | 0.00 | 0.00 |
| Sync-Neg/C | 74.77 | 99.21 | 99.98 | 99.99 | 99.99 | 32.76 | 94.64 | 99.97 | 99.99 | 99.99 | — | — | — | — | — |

Table 24: Significance of profit comparison under variations of nominal load. A table entry gives the probability (in percent) that the policy in the row results in a higher average profit than the policy above the column, given the nominal load of the column.

6.6.2 Variations in degree of uncertainty

We have tested the policies subject to variations in the degree of uncertainty. The degree of uncertainty has been sampled by varying the mean-busy-time-to-failure (\bar{t}_{busy}). We have calculated the corresponding mean resource efficiencies using the equation in Section 6.3.2. A mean resource efficiency of 1.00 corresponds to an infinite mean-busy-time-to-failure, that is, no resource breakdowns. The mean efficiency decreases as the frequency of resource breakdowns increases. Experimental results are reported in Table 25, Figure 42 and Figure 43.

The results indicate, as could be expected, that average profits decrease as resource efficiencies decrease. For low frequencies of resource breakdowns, profit is less sensitive to variations when utilizing synchronization policy than is the case in respect to reference policies. The situation is reversed for higher frequencies, synchronization policy being more sensitive. This behavior may be explained as follows: Synchronization policy absorbs resource breakdowns by the insertion of time buffers. The sizes of these buffers are predefined and do not adapt to changes in the environment. As the frequency of resource breakdowns becomes more and more severe, the time buffers thus lose their ability to absorb uncertainty and the tardiness cost soars. Furthermore, as shown in Figure 43, the lack of ability to anticipate and adjust to the level of uncertainty is also devastating for the quality of the forecasts for completion. Too tight estimates result in the acceptance of too many orders, while too loose estimates result in the rejection of orders for which there may be sufficient capacity. Reference policies, however, have the congenial ability of learning about the impact of resource breakdowns through the use of historical leadtime data. This behavior is also reflected in the performance metrics of Table 25. Synchronization policy rejects more bids than reference policies for high resource efficiencies, but it rejects far fewer bids for low efficiencies. As a consequence, the percentage of tardy orders for the Sync-Neg/C policy increases from 20 to 85 percent, while it remains between 47 and 50 percent for the Lead-Neg/C policy.

For many production environments, the level of resource breakdowns over time may be reasonably stable and the ability to adapt to changes is not that important. However, the experiment indicates that there may be benefits in enhancing the synchronization policy with mechanisms that can learn and react to fluctuations within more unsteady environments. We have completed an experiment that reveals the potential benefits of a learning mechanism. In this experiment we have fine-tuned the time buffers for the individual resource efficiencies. The results are shown as the dotted line in Figure 42. Closer studies of these kinds of mechanisms are beyond the scope of this thesis, but a preliminary discussion is provided in Section 7.2, and some further details about the learning experiment are provided in appendix B.

| Policy | Eff | Bids | R% | Compl | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|------------|------|------------|--------------|------------|------------|--------------|------------|----------|--------------|---------------|--------------|---------------|
| Lead-Neg/C | 1.00 | 804 ±13 | 7.1 ±1.4 | 741 ±18 | 366 ±62 | 87.0 ±3.3 | 140 ±15 | 24 ±2 | 1255±1 58 | 1477 ±805 | 6365 ±153 | 3633 ±993 |
| | 0.91 | 804 ±13 | 13.8 ±1.8 | 686 ±20 | 328 ±62 | 85.8 ±4.2 | 185 ±19 | 43 ±4 | 1509±1 57 | 1909 ±921 | 5824 ±158 | 2405 ±1096 |
| | 0.83 | 804 ±13 | 21.6 ±2.5 | 630 ±21 | 311 ±62 | 85.4 ±5.4 | 224 ±21 | 61 ±4 | 1636±1 56 | 2112 ±1008 | 5311 ±186 | 1563 ±1139 |
| | 0.71 | 804 ±13 | 34.5 ±3.2 | 531 ±25 | 257 ±48 | 78.9 ±6.1 | 260 ±22 | 77 ±3 | 1551±1 27 | 1840 ±899 | 4363 ±209 | 971 ±1002 |
| FCS-Neg/C | 1.00 | 804 ±13 | 11.0 ±0.9 | 711 ±15 | 272 ±47 | 83.5 ±4.4 | 135 ±9 | 41 ±2 | 1162 ±128 | 686 ±386 | 6079 ±126 | 4231 ±521 |
| | 0.91 | 804 ±13 | 16.5 ±1.7 | 671 ±18 | 289 ±50 | 83.5 ±4.7 | 178 ±14 | 52 ±3 | 1417 ±143 | 1217 ±581 | 5687 ±137 | 3052 ±690 |
| | 0.83 | 804 ±13 | 22.1 ±2.6 | 628 ±21 | 297 ±47 | 85.1 ±4.2 | 212 ±15 | 62 ±3 | 1562 ±140 | 1533 ±630 | 5270 ±170 | 2174 ±780 |
| | 0.71 | 804 ±13 | 34.4 ±2.8 | 537 ±22 | 277 ±45 | 81.4 ±6.2 | 261 ±24 | 76 ±3 | 1558 ±190 | 1877 ±803 | 4430 ±184 | 996 ±990 |
| Sync-Neg/C | 1.00 | 804 ±13 | 12.9 ±1.2 | 701 ±15 | 139 ±10 | 79.9 ±3.6 | 114 ±6 | 41 ±3 | 1239 ±95 | 79 ±32 | 5846 ±125 | 4528 ±187 |
| | 0.91 | 804 ±13 | 15.7 ±1.5 | 675 ±17 | 258 ±13 | 83.0 ±3.1 | 142 ±8 | 48 ±4 | 1246 ±89 | 343 ±76 | 5597 ±145 | 4008 ±235 |
| | 0.83 | 804 ±13 | 19.1 ±1.9 | 646 ±18 | 386 ±12 | 84.3 ±3.3 | 173 ±10 | 54 ±4 | 1300 ±103 | 936 ±158 | 5327 ±156 | 3090 ±330 |
| | 0.71 | 804 ±13 | 25.5 ±1.9 | 600 ±20 | 509 ±14 | 84.8 ±3.3 | 242 ±13 | 66 ±4 | 1504 ±112 | 2693 ±253 | 4858 ±176 | 661 ±436 |

Table 25: Performance metrics. The column denoted “Eff” gives the mean resource efficiencies.

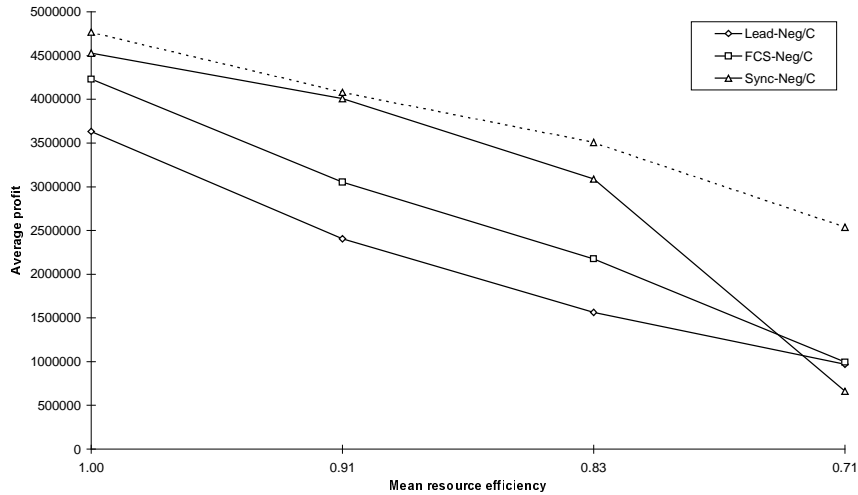


Figure 42: Sensitivity to variations in breakdown frequency. The dotted line represents the Sync-Neg/C policy with the time buffers adjusted for the individual resource efficiencies, and thus representing the potential performance improvements that can be expected from enhancing the Sync-Neg/C policy with a mechanism capable of learning and reacting to long-term changes in the environment.

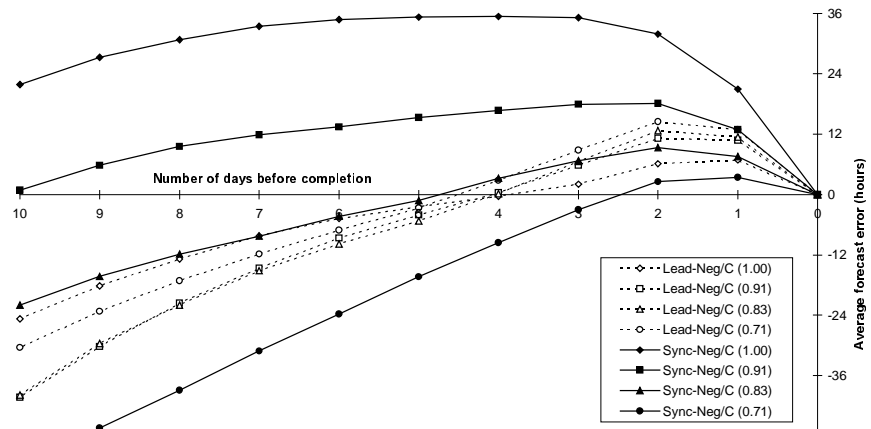


Figure 43: Comparison of average forecast error for the Lead-Neg/C and Sync-Neg/C policies under various resource efficiencies.

7. Conclusions

7.1 Thesis summary and conclusions

This dissertation addresses coordination aspects of supply chain management. The approach taken is mainly targeted to decentralized supply chains in which each entity is responsible for managing its operations while coordinating with both downstream and upstream facilities. Such an environment is typical of supply chains where different entities belong to different companies, but it is also representative of internal (intra-enterprise) supply chains where each facility is managed in a semi-autonomous manner. In fact, as supply chains strive to operate as single extended virtual enterprises, the distinction between intra- and inter-enterprise supply chains are increasingly becoming irrelevant. The decentralized yet coordinated supply chain models assumed in this dissertation are intended to be representative of this shift towards extended virtual enterprises.

Classical supply chain practices can be characterized as relying on reordering policies for inventory buffers to achieve coordination. In light of the trend towards stockless just-in-time production, such practices will become less applicable and more synchronized means of coordination must therefore be developed. The research presented in this dissertation focuses on this issue. We built on the MASCOT agent-based framework first introduced by Sadeh (Sadeh 1996) and specified mechanisms for the coordination of schedule information across the supply chain, as well as a number of coordination policies applicable within this framework. These coordination policies were empirically evaluated using a realistic simulation testbed to account for various sources of the uncertainty that makes supply chain coordination such a challenging problem. The performance metrics utilized for the evaluation were: number of bids that resulted in orders, number of orders completed in time, utilization of bottleneck resources, average leadtime, in-system inventory cost, tardiness cost, revenue, profit, and forecast accuracy.

The supply chain coordination policies defined when, what, and with whom to communicate and how to make use of the information received. These policies were grouped into the two categories of reference policies and synchronization policies. The reference policies used leadtime-based methods to determine internal release- and due-dates within each supply chain entity, and did not exchange information during execution of the orders. A variation of the reference policies

additionally allowed for checking finite capacity constraints in the first tier of the supply chain either to determine whether a bid was to be rejected or specify what its eventual delivery date to the customer might be. The synchronization policies determined, and regularly updated, the internal due dates based on the execution status and available capacity both upstream, downstream, and locally in the supply chain. We started by presenting a set of just-in-time policies, where need dates from customers become internal due dates and promise dates from suppliers become internal release constraints. Subsequent sets of policies included mechanisms for time buffering, bid refusal, and promise date negotiation, as well as a negotiation process for suppliers in competition. Competition was modeled as an increased chance for losing bids that cannot be met, especially for bids with high profit margins.

For the most simplistic set of coordination policies (the just-in-time policies), reference policy outperformed synchronization policy. We identified the main reason for the poor performance of the latter to be its brittleness and inability to learn about contingencies. All the other variations of synchronization policies outperformed their leadtime-based counterparts. The high performances of such synchronization policies were assumed to be mainly due to their ability to let individual supply chain entities prioritize based on updated status information from their supply chain partners. Moreover, we found these synchronization policies capable of applying safety leadtimes more selectively and, therefore, of establishing realistic promise dates without relying on excessive in-system inventories. Furthermore, we found that synchronization policies were able to make fairly accurate forecasts for the completion of orders. In cases where requested delivery dates for bids were assumed to be non-negotiable, these forecasts were used to submit only those bids that were expected to be profitable. Otherwise, forecasts were used to propose alternative and more realistic delivery dates. Even though synchronization policies displayed a tendency to reject more of the incoming bids, thereby generating less revenue than the reference policies, they had a significantly higher due-date performance that consequently resulted in a higher average profit. When testing these policies for high nominal loads, we found the performance of the synchronization policies to be considerably better than that of the reference policies, again mainly due to their accuracy in forecasting order completion. The last series of experiments reported in this dissertation indicated that synchronization policies outperformed reference policies regardless of the level of contingencies that were introduced into the supply chain, although this required adjusting the amount of safety leadtime to the individual levels of uncertainty. In general, the benefits of synchronization appear to increase with the complexity, load, and degree of uncertainty in the supply chain.

7.2 Directions for future research

There are a number of ways in which the research reported in this dissertation could be extended. These include the following issues:

- **Asynchronous coordination** — In Section 3.2 we advocated asynchronous schedule revision as the only workable mode for coordinating schedules between autonomous entities. The experiments presented in this dissertation assume that synchronization policies periodically rebuild schedules that are consistent across the supply chain, each being triggered by an external synchronization controller. They thus represent idealized situations, such as the implicit assumption that a supply chain agent can re-optimize its schedule momentarily. Further refinements of the simulation testbed should support the asynchronous mode of coordination described in Section 4.6. The results presented in this dissertation can then be used as benchmarks for comparison with the performance of asynchronous coordination.
- **Ripple effects** — One question that has not been raised in this dissertation is whether individual policy changes will have ripple effects that transmit to other businesses in a market that then forces the latter follow. The supply chain in Section 6.5.2 (see Figure 35 on page 103) may be used to exemplify this idea. Assuming that Agents 2 and 3 rely on leadtime-based supply chain coordination (e.g., Lead-Neg/C), we may wish to know what effect it will have on the performance of Agent 2 if Agent 3 switches to a synchronization-based policy (e.g., Sync-Neg/C) for coordination with suppliers.¹⁸ This relates to the Nash equilibrium stability criterion presented in Section 3.4.1. Since the current version of the simulation testbed does not allow for a mix of policies within the supply chain, it cannot answer this question. However, we can see no major obstacles to enhancing the testbed to support this kind of environment.
- **Competing supply chains** — Improved customer performance in a business is definitely a competitive advantage that may influence competing businesses through changes in customer preferences. Such questions have been addressed in *game theory* literature (Rosenschein and Zlotkin 1994). We would like to simulate competing supply chains where agents learn the customer satisfaction performance of their suppliers. Enhancing agents by means of such capabilities makes it possible to study the long-term effects of supply chain policies more closely. For

¹⁸ Swaminathan (1996) has shown that exchange of information may not always be beneficial for isolated agents, that is, in some cases Agent 2 may be better off continuing to use a leadtime based coordination policy.

example, a supplier that tends to make promises he cannot keep might profit from this in the short term, but he could also suffer long-term losses due to increased disbelief among customers.

- Production based on sales forecasts — The experiments presented in this dissertation assume no presence of stock or buffers of raw materials or intermediate or finished products. We have gained a certain initial insight through our projects into how stocks could be maintained by dynamic and delayed assignment of supply to demand. One way to allow a supply chain entity to carry stock might be to introduce a separate *forecast agent* for each supply chain entity that is responsible for generating demand based on current stock levels, reordering policies, and estimates of future sales. The forecast agent will consequently generate production orders to meet these demands. When a firm customer order (or an available-to-promise request) arrives, it might first be checked for matching demand in the forecast agent. If a matching demand is found, the corresponding order is re-assigned since otherwise a new production order would be generated. We anticipate only minor enhancements of the framework to support this kind of functionality.
- Adaptive safety leadtime buffers — In Section 6.6.2 we identified situations where mechanisms that allow automated learning and updating of the safety leadtime buffers may be beneficial. The size of these time buffers is critical for forecast accuracy and for available-to-promise quotations. Such mechanisms could evaluate forecast errors for orders as they complete and try to identify sources of any possible deviations. For example, if material was present on time, the duration of all activities according to schedule, and the order still completed later than forecasted, then possible corrective action might be to increase safety leadtime buffers.

References

- Adelsberger, H. and J. Kanet (1991). The Leitstand — A New Tool for Computer Integrated Manufacturing, *Production and Inventory Management Journal*, first quarter, pp. 43-48.
- Allen, James F. (1984). Towards a General Theory of Action and Time. *Artificial Intelligence*, Vol. 23, No. 2, July 1984, 123-154.
- Anand, Krishnan S. and Haim Mendelson (1997). Information and Organization for Horizontal Multimarket Coordination. *Management Science*, 43(12), 1609-1627.
- Anupindi, Ravi (1993). *Supply Management Under Uncertainty*. Ph.D. Thesis, Carnegie Mellon University, August 1993.
- Anupindi, Ravi and Ram Akella (1993). Diversification Under Supply Uncertainty. *Management Science*, 38(8):944-963, August 1993.
- APICS (1987). *APICS Dictionary*. Sixth Edition, (prepared by Thomas F. Wallace and John R. Dougherty), ISBN 0-935406-90-S.
- Arntzen, Bruce C., Gerald G. Brown, Terry P. Harrison, and Linda L. Trafton (1995). Global Supply Chain Management at Digital Equipment Corporation, *Interfaces*, 25(1), 69-93, 1995.
- Baker, Kenneth R. (1992). *Elements of Sequencing and Scheduling*. Amos Tuck School of Business Administration, Dartmouth College, Hanover, NH. 1992.
- Barbuceanu, Mihai and Mark S. Fox (1994). The Information Agent: An Infrastructure Agent Supporting collaborative Enterprise Architectures. *Third Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises*, Morgantown, WV, IEEE Computer Science Press, 1994.
- Bassok, Y. and R. Akella (1991). Ordering and production decisions with supply quality and demand uncertainty. *Management Science*, 37(12), 1556-1574.
- Beck, J. Christopher and Mark S. Fox (1994). Supply Chain Coordination via Mediated Constraint Relaxation. In *Proceedings of the First Canadian Workshop on Distributed Artificial Intelligence*, Banff, AB, May 15, 1994.

- Benjamin, Robert and Rolf Wigand (1995). Electronic Markets and Virtual Value Chains on the Information Superhighway. *Sloan Management Review*, Winter 1995, 62-72.
- Bensana, E., M. Corregge, G. Bel and D. Dubois (1986). An expert-system approach to industrial job shop scheduling. *International Conference on Robotics and Automation*, San Francisco, CA, April 1986, 1645-1650.
- Bensana, E., G. Bel and D. Dubois (1988). OPAL: A multi-knowledge-based system for industrial job shop scheduling. *International Journal of Production Research*, 26(5), 795-819.
- Bielecki, T. and R. Kumar (1988). Optimality of zero-inventory policies for unreliable manufacturing systems. *Operations Research*, 36(4), 1988.
- Blazewicz, J., K. Ecker, G. Schmidt, and J. Weglarz (1993). *Scheduling in Computer and Manufacturing Systems*. Springer-Verlag, 1993.
- Boctor, F. (1990). Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research*, 49:3-13.
- Bond, Alan H. and Les Gasser (1988). *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.
- Borning, Alan, Bjorn Freeman-Benson, Molly Wilson (1992). Constraint Hierarchies. *Lisp and Symbolic Computation*, 5, 233-270, 1992.
- Bredrup, Harald (1995). *Performance Measurements in a Changing Competitive Industrial Environment: Breaking the Financial Paradigm*. Ph.D. Thesis 1995:111, University of Trondheim, November 1995.
- Burbidge, John L. (1975). *The introduction of group technology*. John Wiley & Sons, New York, 1975.
- Burke, Peter and Patrick Prosser (1994). The Distributed Asynchronous Scheduler. In M. Zweben and M. Fox (Eds.), *Intelligent Scheduling*, pp. 309 - 340, Morgan Kaufmann, 1994.
- Cantamessa, Marco (1997). Agent-based modeling and management of manufacturing systems. *Computers in Industry*, 34 (1997), 173-186.
- Chaib-draa, B. (1992). Distributed Artificial Intelligence: An Overview. *Artificial Intelligence Review*, 6(1):35-66, 1992.

- Chiang, W. Y. and M. S. Fox (1990). Protection against uncertainty in a deterministic schedule. *Proceedings Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Islands, May 1990, pp. 184-197.
- Choi, Thomas Y. and Janet L. Hartley (1996). An exploration of supplier selection practices across the supply chain. *Journal of Operations Management*, 14 (1996), 333-343.
- Clark, A. J. and H. Scarf (1960). Optimal policies for a multiechelon inventory problem, *Management Science*, 6 (1960), 475-490.
- Cohen, Morris A. and Hau L. Lee (1988). Strategic analysis of integrated production-distribution system: Models and methods. *Operations Research*, 36(2), 216-228, 1988.
- Colmerauer, Alain (1990). An introduction to PROLOG-III. *Communications of the ACM*, 33(7):69-90, July 1990.
- Currie, Ken and Austin Tate (1991). O-Plan: the open planning architecture. *Artificial Intelligence*, 52(1):49-86, 1991.
- Davis, Randall and Reid G. Smith (1983). Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence*, 20 (1983), 69-109.
- Davis, Tom (1993). Effective Supply Chain Management. *Sloan Management Review*, Summer 1993, 35-46.
- Dechter, R., I. Meiri, and J. Pearl (1991). Temporal Constraint Networks. *Artificial Intelligence*, 49 61-95, Elsevier, 1991.
- Della Croce, F., G. Menga, R. Tadei, M. Cavalotto, and L. Petri (1993). Cellular control of manufacturing systems. *European Journal of Operational Research*, 69, 498-509.
- Dincbas, M., P. van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier (1988). *The Constraint Logic Programming Language CHIP*. Technical Report TR-LP-37, European Computer Industry Research Centre, Munich, Germany, May 1988.
- Dorn, Jürgen and Carl Froeschl (1993). *Scheduling of Production Processes*. Ellis Horwood, 1993.

- Dorn, Jürgen, Roger Kerr, and Gabi Thalhammer (1994). Reactive Scheduling in a Fuzzy-Temporal Framework. In E. Szelke and R. Kerr (Eds.), *Knowledge-Based Reactive Scheduling*, Elsevier Science Publishers.
- Dorn, Jürgen (1995). Iterative Improvement Methods for Knowledge-based Scheduling. *AI Communications*, 8(1), 20-34, 1995.
- Drake, G. R. and J. S. Smith (1996). Simulation system for real-time planning, scheduling, and control. *Proceedings of the 1996 Winter Simulation Conference*, 1083-1090.
- Durfee, Edmund H., Victor R. Lesser, and Daniel D. Corkill (1989). Trends in Cooperative Distributed Problem Solving. *IEEE Transactions*, March 1989.
- van der Duyn Schouten, Frank A., Marc J. G. van Eijs, and Ruud M. J. Heuts (1994). The Value of Supplier Information to Improve Management of a Retailer's Inventory. *Decision Sciences*, 25(1):1-14, January/February 1994.
- Elleby, P., H. E. Fargher, and T. R. Addis (1989). A constraint-based scheduling system for VLSI wafer fabrication. In J. Browne (Ed.), *Knowledge-Based Production Management Systems*, Elsevier Science Publishers, 107-114.
- Erman, Lee D., Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy (1980). The HERSAY-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 12(2):213-253, June 1980.
- Feigenbaum, Edward A. (1977). The art of artificial intelligence: Themes and case studies in knowledge engineering, *The 5th International Joint Conference on Artificial Intelligence*, Cambridge, MA, August 1977, pp. 1014-1029.
- Feigin, G., C. An, D. Connors, and I. Crawford (1996). Shape Up, Ship Out. *OR/MS Today*, 24-30, April 1996.
- Fox, Kenneth. A (1984). MRP-II providing a natural 'hub' for computer-integrated manufacturing systems. *Industrial Engineering*, 16(10), 44-50, 1984.
- Fox, Mark S. (1983). *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. Ph.D. Thesis, Carnegie Mellon University, 1983, Morgan Kaufmann, 1987.
- Fox, Mark S. and Stephen F. Smith (1984). ISIS: A knowledge-based system for factory scheduling. *Expert Systems*, 1(1), 25-49.

- Fox, Mark S. and Michael Gruninger (1994). Ontologies for Enterprise Integration. *Proceedings of the 2nd Conference on Cooperative Information Systems*, Toronto, May 1994.
- Fox, Mark S. and Michael Gruninger (1998). Enterprise Modelling. *AI Magazine*, 1998 (forthcoming).
- Fox, Robert E. (1987). OPT: Leapfrogging the Japanese. *Just-in-time Manufacture*, IFS Ltd., Springer-Verlag, 1987.
- French, Simon (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. John Wiley & Sons, 1982.
- Genesereth, M. R. and S. P. Ketchpel (1994). Software Agents. *Communications of the ACM*, 37(7), 48-53, 1994.
- Ginsberg, Matthew L. (1993). Dynamic Backtracking. *Journal of Artificial Intelligence Research*, 1, pp. 25-46.
- Glover, Fred and Manuel Laguna (1993). Tabu Search. In Colin R. Reeves (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*. 70-150, Blackwell Scientific Publications, Oxford, 1993.
- Gmytrasiewicz, Piotr J. and Edmund H. Durfee (1993). Towards a Theory of Honesty and Truth Among Communicating Autonomous Agents. *Group Decision and Negotiation*, 2:237-258 (Special issue on Distributed Artificial Intelligence), 1993.
- Haavardtun, Johan and Dag Kjenstad (1995). An interactive tool for short-term scheduling. In *Proceedings of The First World Congress on Intelligent Manufacturing Processes & Systems*, 466-477, February 1995.
- Haavardtun, Johan (1995) *A Production Planner's Workbench*. Ph.D. Thesis 1995:75. University of Trondheim, Norway.
- Hall, Robert W. (1983). *Zero Inventories*. Dow Jones-Irwin, 1983, ISBN 0-87094-461-4.
- Harmon, Roy L. (1992). *Reinventing the Factory II*. The Free Press, 1992, ISBN 0-92-913862-0.
- Hastings, N. A. J., and C.-H. Yeh (1990). Job oriented production scheduling. *European Journal of Operational Research*, 47, 35-48.

- Held, Michael and Richard M. Karp (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196-210, March 1962.
- Helper, S. (1991). How much has really changed between US automakers and their suppliers?. *Sloan Management Review*, 32(4), 15-28, 1991.
- Henkoff, R. (1994). Delivering the goods. *Fortune*, Vol. 130, No. 11, E34-47, Nov 28, 1994.
- Hildum, David W., Norman M. Sadeh, Thomas J. Laliberty, Stephen F. Smith, John McA'Nulty, and Dag Kjenstad (1996). Mixed-Initiative Management of Integrated Process-Planning and Production Scheduling Solutions. *Proceedings, AI and Manufacturing Research Planning Workshop*, Albuquerque, NM, June, 1996.
- Hines, William W. and Douglas C. Montgomery (1990). *Probability and Statistics in Engineering and Management Science*. 3rd edition, John Wiley & Sons, 1990.
- Hinkkanen, A., R. Kalakota, P. Saengcharoenrat, J. Stallert, and A. B. Whinston (1997). Distributed Decision Support Systems for Real Time Supply Chain Management using Agent Technologies. In R. Kalakota and A. B. Whinston (Eds.), *Readings in Electronic Commerce*, Addison Wesley, 1997, pp. 275-291.
- Hynynen, Juha (1988). *A framework for coordination in distributed production management*. Ph.D. Thesis, Helsinki University of Technology, May 1988.
- Jacobs, F. Robert (1984). OPT uncovered: Many production planning and scheduling concepts can be applied with or without the software. *Industrial Engineering*, 16(10), 32-41, 1984.
- Jaffar, J., S. Michaylow, P. Stuckey, and R. Yap (1990). *The CLP(\mathcal{R}) language and system*. IBM Research Division, Technical Report RC 16292 (#72336), November 1990.
- Jain, S., K. Barber, and D. Osterfeld (1989). Expert simulation for on-line scheduling. *Proceedings of the 1996 Winter Simulation Conference*, 930-935.
- Japan Management Association (1986). *Kanban and just-in-time at Toyota; management begins at the workplace*. Productivity Press, 1986.

- Johnson, S. M. (1954). Optimal Two- and Three-Stage Production Schedules with Setup Times Included. *Naval Research Logistics Quarterly*, 1(1):61-68, March 1954.
- Kalakota, Ravi, Jan Stallaert, and Andrew B. Whinston (1995). Implementing Real-time Supply Chain Optimization Systems. *Proceedings of the Conference on Supply Chain Management*, Hong Kong, 1995.
- Kerr, R. M. and R. W. Ebsary (1988) Implementation of an expert system for production scheduling. *European Journal of Operational Research*, 33, 17-29.
- Kerr, R. M. and R. N. Walker (1989). A Job Shop Scheduling System Based on Fuzzy Arithmetic. *Proceedings of the Third International Conference on Expert Systems and the Leading Edge in Production and Operations Management*, Hilton Head Island, SC, 433-450, May 1989.
- Knill, Bernie (1994). Quick Response too slow for the 90's. *Industry Week*, 243(9), S16-21, 1994.
- Kolodner, Janet L., Robert L. Simpson Jr., and Katia Sycara-Cyranski (1985). A Process Model of Case-Based Reasoning in Problem Solving. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles CA, August 1985, Vol. 1, 283-290.
- Kraus, Sarit, Jonathan Wilkenfeld, and Gilad Zlotkin (1995). Multiagent negotiation under time constraints. *Artificial Intelligence*, 75 (1995), 297-345.
- Kumar, A., P. S. Ow, and M. J. Prietula (1993). Organizational Simulation and Information System Design: An Operations Level Example. *Management Science*, 39(2), 218-240, 1993.
- Kunnathur, A. S., S. Sampath, and P. S. Sundararaghavan (1996). Dynamic Rescheduling of a Job Shop: A Simulation Study. *Proceedings of the 1996 Winter Simulation Conference*, 1091-1098.
- Kurtulus, I. and E. W. Davis (1982). Multi-project scheduling: Categorization of heuristic rules performance. *Management Science*, 28(2), 1982, 161-172.
- van Laarhoven, Peter J. M., Emile H. L. Aarts, and Jan Karel Lenstra (1992). Job Shop Scheduling by Simulated Annealing. *Operations Research*, 40(1), 113-125.

- Laborie, Philippe and Malik Ghallab (1995). Planning with Sharable Resource Constraints. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1643-1649.
- Lau, Hon-Shiang and Amy Hing-Ling Lau (1994). Coordinating two suppliers with offsetting lead time and price performance, *Journal of Operations Management*, 11, 327-337, 1994.
- Law, Averill M. and W. David Kelton, (1992). *Simulation Modeling & Analysis*. Second Edition, McGraw-Hill.
- Lawler, E. L. and D. E. Wood (1966). Branch and Bound Methods: A Survey. *Operations Research*, 14(4):699-719, July-August 1966.
- Leavy, Brian (1994). Two strategic perspectives on the buyer-supplier relationship. *Production and Inventory Management Journal*, 35(2):47-51.
- Lee, Hau L. and Corey Billington (1992). Managing Supply Chain Inventory: Pitfalls and Opportunities. *Sloan Management Review*, Spring 1992.
- Lee, Hau L. and Corey Billington (1993). Material management in decentralized supply chains. *Operations Research*, 41(5), 835-847, 1993.
- Le Pape, Claude (1991). *Constraint Propagation in Planning and Scheduling*. Technical Report, Stanford University, 1991.
- Le Pape, Claude (1994). Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems. *Intelligent Systems Engineering*, 1994.
- Le Provost, Thierry and Mark Wallace (1992). *Generalized constraint propagation over the clp scheme*. Technical Report ECRC-92-1, European Computer Industry Research Centre, Munich, Germany.
- Lin, Grace Y.-J. and James J. Solberg (1992). Integrated shop floor control using autonomous agents. *IIE Transactions*, 24(3):57-71, 1992.
- Liu, Jyi-Shane (1996). *Coordination of Multiple Agents in Distributed Manufacturing Scheduling*. Ph.D. Thesis, Carnegie Mellon University, 1996.
- Lyons, Thomas F., A. Richard Krachenberg, and John W. Henke Jr. (1990). Mixed motive marriages: What's next for buyer-supplier relations?. *Sloan Management Review*, 29-36, Spring 1990.

- Maes, Pattie (1994). Modeling Adaptive Autonomous Agents. *Artificial Life Journal*, Vol. 1, No. 1&2, 135-162.
- Malone, Thomas W. (1987). Modeling Coordination in Organizations and Markets. *Management Science*, 33(10), 1317-1332, 1987.
- Malone, Thomas W. and Kevin Crowston (1994). The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, Vol. 26, No. 1, March 1994, 89-119.
- Mandell, M. (1991a). Kmart's \$1 billion bar-code bet. *Computerworld*, August 12, 1991, pp. 53-55.
- Mandell, M. (1991b). EDI speeds Caterpillar's global march. *Computerworld*, August 12, 1991, p. 58.
- Mattfeld, Dirk C. (1995). *Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling*. Heidelberg: Physica-Verlag.
- Maturana, Francisco P. and Douglas H. Norrie (1995). *A Generic Mediator for Multi-Agent Coordination in a Distributed Manufacturing System*. Division of Manufacturing Engineering, Department of Mechanical Engineering, University of Calgary, <http://ksi.cpsc.ucalgary.ca/projects/Mediator/>.
- Meng, Chao-Chiang and Michael Sullivan (1991). LOGOS: A constraint-directed reasoning shell for operations management, *IEEE Expert*, 6(1):20-28, February 1991.
- Miller, Robert W (1963). *Schedule, cost, and profit control with PERT: A comprehensive guide for program management*. McGraw-Hill, New York.
- Miyashita, Kazuo (1994). *A Case-Based Approach to Improve Quality and Efficiency in Ill-Structured Optimization: An Application to Job Shop Scheduling*. Ph.D. Thesis, Department of Electronic Engineering, Faculty of Engineering, Osaka University, November 1994.
- Miyashita, Kazuo and Katia Sycara (1994). Adaptive case-based control of schedule revision. In M. Zweben and M. Fox (Eds.), *Intelligent Scheduling*, pp. 291-308, Morgan Kaufmann, 1994.
- Miyashita, Kazuo (1998). CAMPS: A Constraint-Based Architecture for Multiagent Planning and Scheduling. *Journal of Intelligent Manufacturing*, Vol. 9, 145-154.

- Morton, Thomas E. and David W. Pentico (1993) *Heuristic Scheduling Systems: with Applications to Production Systems and Project Management*. John Wiley & Sons, New York, 1993.
- Muscettola, N. and S. F. Smith (1987). A probabilistic framework for resource-constrained multi-agent planning. *Proceedings 10th International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 1063-1066.
- Muscettola, N. and S. F. Smith (1990). Integrating planning and scheduling to solve space mission scheduling problems. *Proceedings 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, San Diego, CA, November 1990, pp. 220-230.
- Muscettola, N., S. Roth, N. Sadeh, and K. Sycara (1994). *Knowledge-based logistics planning: Its application in manufacturing and logistics planning*. Technical Report RL-TR-94-206, Rome Laboratory, New York.
- Nakakuki, Yoichiro and Norman M. Sadeh (1994). Increasing the efficiency of simulated annealing by learning to recognize (un)promising runs. In *Proceedings of the 12th National Conference on Artificial Intelligence*, 1316-1322, 1994.
- Narayanan, S. (1994). ECR gets GMA response, *Retail World*, 47(21), page 14, October 24, 1994.
- Nash, John (1951). Non-cooperative games. *Annals of Mathematics*, Vol. 54, No. 2, September 1951.
- NEVEM working group (1989). *Performance indicators in logistics*. IFS Publications, Springer-Verlag, 1989.
- Nowicki, E. and C. Smutnicki (1996). A Fast Taboo Search Algorithm for the Job Shop Problem. *Management Science*, 42(6), 797-813.
- Nuijten, W. P. H. (1994). *Time and Resource Constrained Scheduling. A Constraint Satisfaction Approach*. Doctoral Thesis, Technische Universiteit Eindhoven, December 1994.
- Nwana, H. (1996). Software Agents: An Overview. *Knowledge Engineering Review*, 11(3), 205-244, October/November 1996.
- Ohno, Taiichi (1988). *Toyota Production Systems: Beyond large-scale production*. Productivity Press, 1988.

- Ow, Peng Si, Stephen F. Smith, and Alfred Thiriez (1988a). Reactive Plan Revision. In *Proceedings of the 7th National Conference on Artificial Intelligence*, 77-82, 1988.
- Ow, Peng Si, Stephen F. Smith, and Randy Howie (1988b). A cooperative scheduling system. In Michael D. Oliff (Ed.), *Expert Systems and Intelligent Manufacturing*, 43-56, North-Holland, New York.
- Panwalkar S. S. and Wafik Iskander (1977). A Survey of Scheduling Rules. *Operations Research*, Vol. 25, No. 1, 45-61, January-February 1977.
- Patterson, James H. (1984). A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. *Management Science*, 30(7):854-867, July 1984.
- Porter, Michael E. (1980). *Competitive Strategy: Techniques for Analyzing Industries and Competitors*. New York: Free Press, 1980.
- Pritsker, A Alan B., Lawrence J. Watters, and Philip M. Wolfe (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1), 93-108, September 1969.
- Pyke, David F. and Morris A. Cohen (1993). Performance characteristics of stochastic integrated production-distribution systems. *European Journal of Operations Research*, 68(1), 23-48, 1993.
- Reid, Michael (1995). Stores of value. *The Economist*, London, March 4, 1995.
- Rit, Jean-Francois (1986). Propagating temporal constraints for scheduling. In *Proceedings of the 5th National Conference on Artificial Intelligence*, Vol. 1, pp. 383-388.
- Rolstadås, Asbjørn. (1995). Production Planning in the Virtual Enterprise. In *Proceedings of The First World Congress on Intelligent Manufacturing Processes & Systems*, 779-789, February 1995.
- Romero, Bernardo Prida (1991). The other side of JIT in supply management. *Production and Inventory Management Journal*, 32(4):1-2.
- Rosenschein, Jeffrey S. and Gilad Zlotkin (1994). *Rules of Encounter*. The MIT Press, Cambridge, Massachusetts, ISBN 0-262-18159-2.
- Sadeh, Norman (1991). *Look-Ahead Techniques for Micro-Opportunistic Job-Shop Scheduling*. Ph.D. Thesis, Carnegie Mellon University, March 1991.

- Sadeh, Norman and Stephen F. Smith (1993). *Knowledge-Based Supply Chain Management: An Overview of Ongoing Research at Carnegie Mellon University*. Intelligent Coordination and Logistics Laboratory, Carnegie Mellon University, Internal Report, 1993.
- Sadeh, Norman (1994). Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler. In M. Zweben and M. Fox (Eds.), *Intelligent Scheduling*, pp. 99 - 136, Morgan Kaufmann, 1994.
- Sadeh, Norman, Katia Sycara, and Yalin Xiong (1995). Backtracking techniques for the job shop scheduling constraint satisfaction problem. *Artificial Intelligence*, 76, 455-480.
- Sadeh, Norman (1996). *MASCOT An Agent Architecture for Multi-Level Mixed Initiative Supply Chain Coordination*. Intelligent Coordination and Logistics Laboratory, Carnegie Mellon University, Internal Report.
- Sadeh, Norman M., David W. Hildum, Dag Kjenstad, Allen Tseng, Thomas J. Laliberty, and John McA'Nulty (1998). A Blackboard Architecture for Integrated Process Planning and Production Scheduling. *Concurrent Engineering: Research & Applications*, Vol. 6, No. 2, June 1998.
- Sandholm, Tuomas W. (1996). *Negotiation among self-interested computationally limited agents*. Ph.D. Thesis, Department of Computer Science, University of Massachusetts Amherst, September 1996.
- Sen, Sandip and Edmund H. Durfee (1994). The role of commitment in cooperative negotiation. *International Journal of Intelligent and Cooperative Information Systems*, 3(1):67-81, 1994.
- Shaw, M. J. (1988). Knowledge-based scheduling in flexible manufacturing systems: An integration of pattern-directed inference and heuristic search, *International Journal of Production Research*, 26(5):821-844.
- Simonis, Helmut and Trijntje Cornelissens (1995). Modelling Producer/Consumer Constraints. *Principles and Practice of Constraint Programming*, 1995, 449-462.
- Smith, Reid G. (1980). The contract net protocol: High-level communication and control in the distributed problem solver. *IEEE Transactions on Computers*, Vol. C-29, No. 12, December 1980, 1104-1113.

- Smith, Reid G. and Randall Davis (1988). Framework for Cooperation in Distributed Problem Solving. In A. H. Bond and L. Gasser (Eds.), *Readings in Distributed Artificial Intelligence*, 61-70, Morgan Kaufmann Publishers, San Mateo, CA, 1988.
- Smith, Stephen F. and Juha E. Hynynen (1987). Integrated decentralization of production management: an approach for factory scheduling. *Proceedings ASME Annual Winter conference: Symposium on Integrated and Intelligent Manufacturing*, Boston, MA, December 1987, pp. 427-440.
- Smith, Stephen F. (1989). *The OPIS Framework for Modeling Manufacturing Systems*. Technical Report CMU-RI-TR-89-30, Carnegie Mellon University.
- Smith, Stephen F. (1992). Knowledge-based production management: approaches, results and prospects. *Production Planning & Control*, 1992, 3(4), pp. 350-380.
- Smith, Stephen F., Ora Lassila, and Marcel Becker (1996). Configurable, Mixed-Initiative Systems for Planning and Scheduling. In A. Tate (Ed.), *Advanced Planning Technology*, the AAAI Press, Menlo Park, CA., USA, May 1996.
- Soares, Carlos (1994). *Evolutionary Computation for the Job-Shop Scheduling Problem*. Minho University, Braga, Portugal, December 1, 1994.
- Solberg, J. J. (1989). Production Planning and Scheduling in CIM. *Proceedings of the 11th World Computer Congress, IFIP*, San Francisco, CA, August 1989.
- Srinivasan, K., S. Kekre, and T. Mukhopadhyay (1994). Impact of Electronic Data Interchange Technology on JIT Shipments. *Management Science*, 40(10), 1291-1304.
- Sun Microsystems (1993). Sun Microsystems' manual page for the *erand48* C Library function for SunOS 5.5.
- Suresh, V. and Djipak Chaudhuri (1993). Dynamic Scheduling — A survey of research. *International Journal of Production Economics*, 32, pp. 53-63.
- Svoronos, A. and P. Zipkin (1991). Evaluation of one-for-one replenishment policies for multiechelon inventory systems. *Management Science*, 37(1), 68-83.
- Swaminathan, Jayashankar M. (1996). *Quantitative analysis of emerging practices in supply chains*. Ph.D. Thesis, Graduate School of Industrial Administration, Carnegie Mellon University, August 1996.

- Sycara, Katia, Steven F. Roth, Norman Sadeh, and Mark S. Fox (1991). Resource Allocation in Distributed Factory Scheduling. *IEEE Expert*, 6(1):20-28, February 1991.
- Sycara, Katia, Keith Decker, Anandee Pannu, Mike Williamson, and Dajun Zeng (1996). Distributed Intelligent Agents. *IEEE Expert*, December 1996.
- Syswerda, Gilbert and Daniel Cerys (1990). Knowledge-based genetic search in schedule optimization. *Proceedings Fourth International Conference on Expert Systems in Production and Operations Management*, Hilton Head Islands, May 1990, pp. 125-133.
- Szelke, Elizabeth and Roger Kerr (1994). Knowledge-based reactive scheduling. *Production Planning & Control*, 1994, 5(2), pp. 124-145.
- Taillard, E. (1994). Parallel taboo search technique for the job shop scheduling problem. *ORSA Journal on Computing*, Vol. 6, No. 2, 108-117, 1994.
- Tate, Austin and Brian Drabble (1995). *O-Plan Architecture Guide Version 2.3*. Artificial Intelligence Applications Institute, University of Edinburgh, United Kingdom, <http://www.ai.ai.ed.ac.uk/~oplan/>.
- Tharumarajah, A. and R. Bemelman (1997). Approaches and issues in scheduling a distributed shop-floor environment. *Computers in Industry*, 34(1997), 95-109.
- Thierry, C, G. Bel, and P. Esquirol (1993). Multi-site scheduling: a constraint based approach. *International Conference on Industrial Engineering and Production Management*, Fucam Mons, June 1993.
- Thomas, Douglas J. and Paul M. Griffin (1996). Coordinated supply chain management. *European Journal of Operational Research*, 94 (1996), 1-15.
- Tönshoff, Hans-K., Jan C. Aurich, and Max S. Winkler (1995). On the way to Autonomous and Cooperative Manufacturing Systems. In *Proceedings of The First World Congress on Intelligent Manufacturing Processes & Systems*, 1388-1399, February 1995.
- Towill, D. R., M. M. Naim, and J. Wikner (1992). Industrial Dynamics Simulation Models in the Design of Supply Chains. *International Journal of Physical Distribution and Logistics Management*, 22(5), 3-13, 1992.
- Tzafestas, S. and G. Kapsiotis (1994). Coordinated control of manufacturing/supply chains using multi-level techniques. *Computer Integrated Manufacturing Systems*, 7(3), 206-212.

- Upton, David M. and Andrew McAfee (1996). The Real Virtual Factory. *Harvard Business Review*, July-August 1996, 123-133.
- Vickrey, William (1961). Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, Vol. 16, 1961, pp. 8-37.
- Wilkins, D. (1989). *Can AI planners solve practical problems?* SRI International, Technical Note 468R, November 1989.
- Womack, James P., Daniel T. Jones, and Daniel Roos (1990). *The Machine That Changed the World*. Rawson Associates, 1990, ISBN 0-89256-350-8.
- Wooldridge, M. and N. R. Jennings (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115-152.
- Wyman, F. P. (1991). Common Features of Simulation Based Scheduling. *Proceedings of the 1996 Winter Simulation Conference*, 341-347.
- Zweben, Monte and Mark Fox (Eds.) (1994). *Intelligent Scheduling*. Morgan Kaufmann Publishers, San Francisco, CA, ISBN 1-55860-260-7.

Appendix A: Parameter settings for the experiments

Problem generator parameters

A supply chain model for the problem generator is read from a text file. Table 26 presents the grammar of the input file in Extended Backus Naur Form (EBNF). An EBNF grammar consists of a set of rules for defining a set of strings (called a language). By expanding the top-level node into its lowest-level terminal constituents, a string that is part of the language can be obtained. Strings are enclosed in double quotes (“...”). Alternative patterns are separated by vertical lines (...|...). Elements enclosed in curled brackets ({...}) may appear zero or more times.

| <i>Rules</i> | | <i>Notes</i> |
|------------------------|--|--------------|
| model ::= | { marginal-cost-info prod-info resource-info plan-info op-dur-info } { tier-info } | |
| marginal-cost-info ::= | “marginal_inventory_cost” mic_{prod}^{\min} mic_{prod}^{\max} | (1) |
| prod-info ::= | “number_of_products” n_{prod} | (1) |
| resource-info ::= | “number_of_resources” n_{res} | (1) |
| plan-info ::= | “operations_per_plan” n_{op}^{\min} n_{op}^{\max} | (1) |
| op-dur-info ::= | “unit_dur” u_{op}^{\min} u_{op}^{\max} | (1) |
| tier-info ::= | “tier:” { prod-info bom-info } { agent-info } “:tier” | (2) |
| bom-info ::= | “number_of_raw” n_{BOM}^{\min} n_{BOM}^{\max} | (1) |
| agent-info ::= | “agent:” { resource-info plan-info op-dur-info } { resource-exception } “:agent” | (3) |
| resource-exception ::= | “resource” integer op-dur-info | (4) |

Table 26: Grammar for the input to the problem generator.

Notes:

- (1) Parameters are defined in Table 4.
- (2) Specifies a tier in the supply chain. Tiers are specified in the sequence given by the material flow. Tier attributes override model attributes.
- (3) Specifies an agent in the tier. Agent attributes override tier and model attributes.
- (4) Specifies exceptional resources where the processing time differs from the default for the agent (e.g., bottleneck resources). The integer number identifies the resource for which the exception applies.

| | |
|---------------------|------|
| number_of_resources | 5 |
| number_of_products | 20 |
| operations_per_plan | 5 5 |
| unit_dur | 1 13 |
| tier: | |
| agent: | |
| resource 1 unit_dur | 1 21 |
| :agent | |
| :tier | |
| tier: | |
| agent: | |
| resource 1 unit_dur | 1 21 |
| :agent | |
| :tier | |

Table 27: Parameters for generating the simple two-tier model used in Section 6.4.

| | |
|---------------------|------|
| number_of_resources | 5 |
| number_of_products | 20 |
| operations_per_plan | 5 5 |
| unit_dur | 1 13 |
| tier: | |
| agent: | |
| resource 1 unit_dur | 1 21 |
| :agent | |
| :tier | |
| tier: | |
| agent: | |
| :agent | |
| :tier | |
| tier: | |
| agent: | |
| resource 1 unit_dur | 1 21 |
| resource 2 unit_dur | 1 21 |
| :agent | |
| :tier | |
| tier: | |
| agent: | |
| :agent | |
| :tier | |
| tier: | |
| agent: | |
| resource 1 unit_dur | 1 21 |
| :agent | |
| :tier | |

Table 28: Parameters for generating the five-tier model used in Section 6.5.1.

```

number_of_resources      5
number_of_products      20
operations_per_plan     5 5
tier:
  agent:
    unit_dur             1 13
    resource 1 unit_dur  1 21
  :agent
:tier
tier:
  agent:
    unit_dur             1 25
    resource 1 unit_dur  1 40
  :agent
  agent:
    unit_dur             1 25
    resource 1 unit_dur  1 40
  :agent
:tier

```

Table 29: Parameters for generating the model used in Section 6.5.2.

```

number_of_resources      5
number_of_products      20
operations_per_plan     5 5
tier:
  agent:
    unit_dur             1 25
    resource 1 unit_dur  1 40
  :agent
  agent:
    unit_dur             1 25
    resource 1 unit_dur  1 40
  :agent
:tier
tier:
  agent:
    unit_dur             1 13
    resource 1 unit_dur  1 21
  :agent
:tier

```

Table 30: Parameters for generating the model used in Section 6.5.3.

| | | | |
|----------------------------|------|---------------------|------|
| number_of_products | 24 | tier: | |
| tier: | | number_of_raw | 1 2 |
| agent: | | agent: | |
| number_of_resources | 4 | number_of_resources | 5 |
| operations_per_plan | 3 4 | operations_per_plan | 5 5 |
| unit_dur | 1 18 | unit_dur | 1 13 |
| resource 1 unit_dur | 8 18 | resource 1 unit_dur | 1 18 |
| :agent | | :agent | |
| agent: | | agent: | |
| number_of_resources | 5 | number_of_resources | 5 |
| operations_per_plan | 5 5 | operations_per_plan | 5 5 |
| unit_dur | 1 15 | unit_dur | 1 14 |
| resource 1 unit_dur | 1 22 | resource 1 unit_dur | 1 16 |
| :agent | | :agent | |
| agent: | | agent: | |
| number_of_resources | 5 | number_of_resources | 3 |
| operations_per_plan | 3 5 | operations_per_plan | 3 3 |
| unit_dur | 1 16 | unit_dur | 1 15 |
| resource 1 unit_dur | 1 23 | resource 1 unit_dur | 1 20 |
| :agent | | :agent | |
| agent: | | :tier | |
| number_of_resources | 7 | tier: | |
| operations_per_plan | 4 7 | number_of_raw | 1 2 |
| unit_dur | 1 15 | agent: | |
| resource 1 unit_dur | 1 24 | number_of_resources | 10 |
| :agent | | operations_per_plan | 2 8 |
| agent: | | unit_dur | 1 20 |
| number_of_resources | 6 | resource 1 unit_dur | 1 34 |
| operations_per_plan | 3 5 | :agent | |
| unit_dur | 1 20 | agent: | |
| resource 1 unit_dur | 1 26 | number_of_resources | 10 |
| resource 2 unit_dur | 1 24 | operations_per_plan | 2 8 |
| :agent | | unit_dur | 1 30 |
| :tier | | :agent | |
| (continued in next column) | | :tier | |

Table 31: Parameters for generating the larger model used in Section 6.5.4 and 6.6.

Simulation parameters

Parameters for the simulation testbed are read from a text file. Table 32 presents the grammar of the input file in EBNF form. The parameters in Table 33 to Table 39 are given according to this grammar.

| <i>Rules</i> | <i>See page</i> |
|---|-----------------|
| simulation-model ::= { scc-padding-info lt-padding-info due-date-info order-size-info bid-frequnecy-info jit-penalty-info revenue-info actual-dur-info mbttf-info mttr-info reject-info } | |
| scc-padding-info ::= “padding” $\alpha \beta s_{max}$ | 144 |
| lt-padding-info ::= “history_percentile” β | 68 |
| due-date-info ::= “avg_due_date” \bar{t}_R | 84 |
| order-size-info ::= “order_size” $q_{bid}^{\min} q_{bid}^{\max}$ | 84 |
| bid-frequnecy-info ::= “bids_per_period” \bar{n}_{bid} | 84 |
| jit-penalty-info ::= “jit_margin” $t_{profitable}$ | 73 |
| revenue-info ::= “revenue” $p_{bid}^{\min} p_{bid}^{\max}$ | 84 |
| actual-dur-info ::= “duration_range” d_{range} | 84 |
| mbttf-info ::= “mean_busy_time_to_failure” \bar{t}_{busy} | 84 |
| mttr-info ::= “mean_time_to_repair” \bar{t}_{repair} | 84 |
| reject-info ::= “reject_limit” c_{rej} | 73 |

Table 32: Grammar for the input to the simulator.

| | |
|---------------------------|-------------|
| avg_due_date | 10080 |
| duration_range | 0.5 |
| history_percentile | 0.7 |
| jit_margin | 10080 |
| mean_busy_time_to_failure | 7200 |
| mean_time_to_repair | 720 |
| order_size | 1 49 |
| bids_per_period | 4 |
| padding | 6 1100 5000 |
| reject_limit | 1000 |
| revenue | 300 400 |

Table 33: Simulation parameters used in Section 6.4.1, 6.4.2 and 6.4.3.

| | |
|---------------------------|------------|
| avg_due_date | 10080 |
| duration_range | 0.5 |
| history_percentile | 0.95 |
| jit_margin | 10080 |
| mean_busy_time_to_failure | 7200 |
| mean_time_to_repair | 720 |
| order_size | 1 49 |
| bids_per_period | 4 |
| padding | 6 800 1440 |
| reject_limit | 1000 |
| revenue | 300 400 |

Table 34: Simulation parameters used in Section 6.4.4 and 6.4.5.

| | |
|---------------------------|-----------|
| avg_due_date | 14400 |
| duration_range | 0.5 |
| history_percentile | 0.95 |
| jit_margin | 10080 |
| mean_busy_time_to_failure | 7200 |
| mean_time_to_repair | 720 |
| order_size | 1 49 |
| bids_per_period | 4 |
| padding | 4 700 720 |
| reject_limit | 1500 |
| revenue | 300 400 |

Table 35: Simulation parameters used in Section 6.5.1.

| | |
|---------------------------|-------------|
| avg_due_date | 10080 |
| duration_range | 0.5 |
| history_percentile | 0.95 |
| jit_margin | 10080 |
| mean_busy_time_to_failure | 7200 |
| mean_time_to_repair | 720 |
| order_size | 1 49 |
| bids_per_period | 4 |
| padding | 6 1000 1440 |
| reject_limit | 1000 |
| revenue | 300 400 |

Table 36: Simulation parameters used in Section 6.5.2 and 6.5.3.

| | |
|---------------------------|-----------|
| avg_due_date | 10080 |
| duration_range | 0.5 |
| history_percentile | 0.95 |
| jit_margin | 10080 |
| mean_busy_time_to_failure | 7200 |
| mean_time_to_repair | 720 |
| order_size | 1 49 |
| bids_per_period | 8 |
| padding | 6 800 720 |
| reject_limit | 1000 |
| revenue | 300 400 |

Table 37: Simulation parameters used in Section 6.5.4.

| | |
|---------------------------|------------|
| avg_due_date | 10080 |
| duration_range | 0.5 |
| history_percentile | 0.95 |
| jit_margin | 10080 |
| mean_busy_time_to_failure | 7200 |
| mean_time_to_repair | 720 |
| order_size | 1 49 |
| bids_per_period | {4,6,8,10} |
| padding | 6 800 720 |
| reject_limit | 1000 |
| revenue | 300 400 |

Table 38: Simulation parameters used in Section 6.6.1. The parameter “bids_per_period” was sampled and is therefore given as a set of values.

| | |
|---------------------------|-----------------------------|
| avg_due_date | 10080 |
| duration_range | 0.5 |
| history_percentile | 0.95 |
| jit_margin | 10080 |
| mean_busy_time_to_failure | { ∞ ,7200,3600,1800} |
| mean_time_to_repair | 720 |
| order_size | 1 49 |
| bids_per_period | 8 |
| padding | 6 1000 2000 |
| reject_limit | 1000 |
| revenue | 300 400 |

Table 39: Simulation parameters used in Section 6.6.2. Parameter “mean_busy_time_to_failure” was sampled and is, therefore, given as a set of values. The fine tuning experiment for the Sync-Neg/C policy used parameter settings for s_{max} according to Table 40.

Appendix B: Auxiliary experiments

This appendix presents experimental results of subordinate or indirect importance for the thesis that, nevertheless, have been necessary in order to set up the major experiments properly.

Percentile fine-tuning for the Buf-Lead policy

The Buf-Lead policy relies on a percentile parameter to determine the amount of safety leadtime to add to the leadtime estimate of a job. Figure 44 shows the performance for sample values of this parameter.

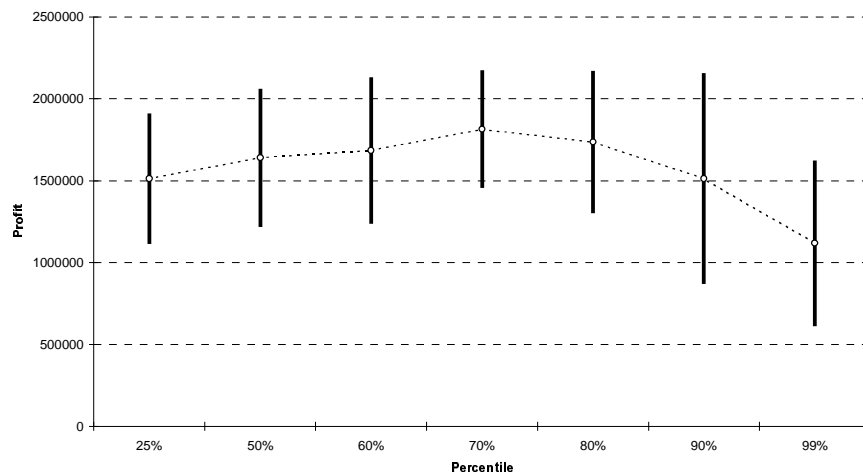


Figure 44: Sampling of percentile for the Buf-Lead policy applied to the simple two-tier model. The high-low-lines indicate the 95 percent confidence intervals for the mean profit. Setting the percentile to 70 percent gives the optimal safety leadtime with respect to profit.

Selection of time buffer function for the Buf-Sync policy

Intuitively, there will be more unpredictable events to handle the farther into the future we look. Therefore, a first safety leadtime approach is to size the time buffers proportional to how far in the future the event is expected to take place (linear padding). At the same time, jobs that are scheduled to be executed far enough in the future tend not to be affected by current unpredictable events as much as jobs that are close to being executed. A second safety leadtime approach is then to try a constant size time buffer (constant padding). Finally, a third approach is to use an “S-shaped”

padding function. This returns only a small time buffer for values close to zero, while for large values there is asymptotic converge to a maximum buffer size. A function given as

$$s(t) = s_{\max} \cdot \text{Gamma}(t, \alpha, \beta)$$

has this property. The parameter s_{\max} is a parameter that controls maximum buffer size (at infinite), Gamma is the cumulative gamma function, and α and β are parameters that define the shape of the cumulative gamma function.

Figure 45 shows the shapes of these three safety leadtime functions. Figure 46 shows the corresponding forecast errors from simulating the 20 experiments of the simple two-tier model. The three safety leadtime functions result in approximately the same average profit, but the S-shaped function gives a much smaller average forecast error. We have thus selected to use the S-shaped padding function in the experiments carried out in Chapter 6. Figure 47 shows average forecast errors using the S-shaped function and indicates the forecast error ranges between “plus one standard deviation” and “minus one standard deviation.” The range of errors is large. For example, five days before completion there is a 50 percent chance that a job will complete later than predicted and a 15 percent chance that the job will complete more than 40 hours later than predicted. This creates an incentive to increase the size of the time buffers so that a higher percentage of promises will be kept. Therefore, in the experiments carried out in Chapter 6, we have selected to use time buffers that optimize profit instead of minimizing forecast errors.

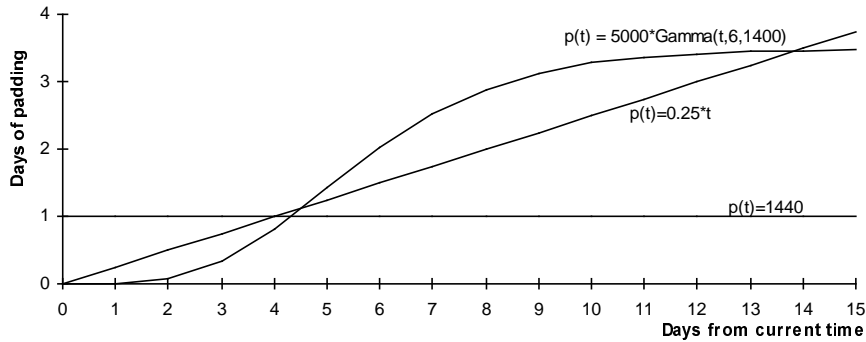


Figure 45: The three safety leadtime functions tested.

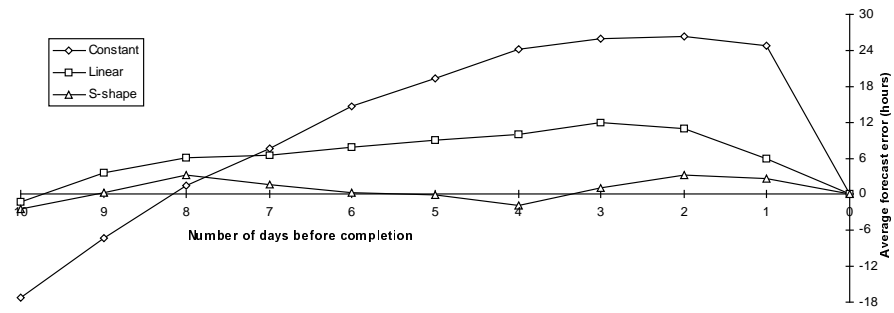
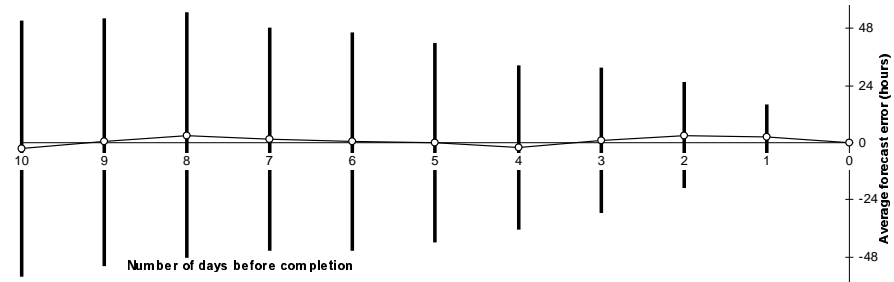


Figure 46: Forecast errors.

Figure 47: Forecast error ranges (mean \pm one standard deviation) with S-shaped safety leadtime buffers.

Percentile fine-tuning for the Lead-Neg policy

The Lead-Neg policy relies on a percentile parameter to determine the appropriate amount of safety leadtime to add to the leadtime estimate of a job. The optimal parameter setting is likely to differ from the one found for the Buf-Lead policy since the Lead-Neg policy allows due dates to be adjustment based on the leadtime estimates. Figure 48 shows the performance for sample values of the percentile parameter.

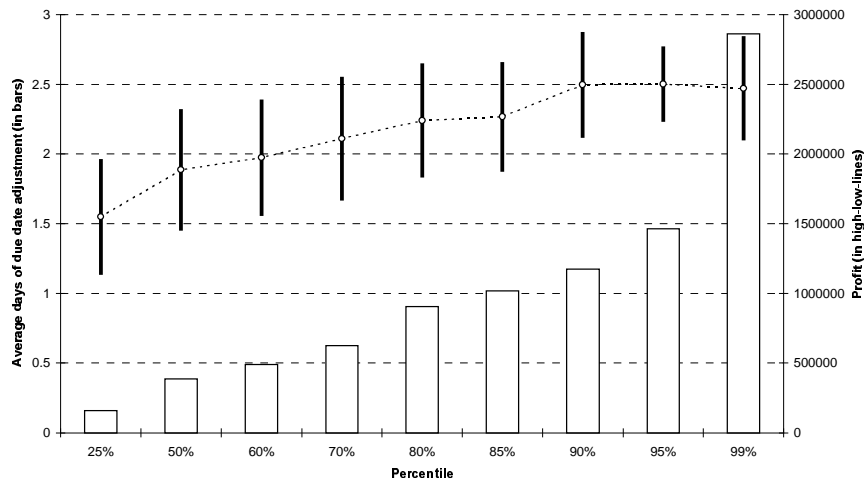


Figure 48: Sampling of percentile for the Lead-Neg policy applied to the simple two-tier model. Setting the percentile to 95 percent gives the optimal safety leadtime with respect to profit.

Percentile fine-tuning for the Lead-Neg/C policy

The Lead-Neg/C policy relies on a percentile parameter to determine the appropriate amount of safety leadtime to add to the leadtime estimate of a job. The competition aspect of the policy might alter the optimal parameter setting of the percentile parameter from the one found for the Buf-Lead policy. We have, therefore, again simulated execution of the 20 randomly generated experiments for sample values of the percentile parameter. Figure 49 shows the corresponding performance metrics.

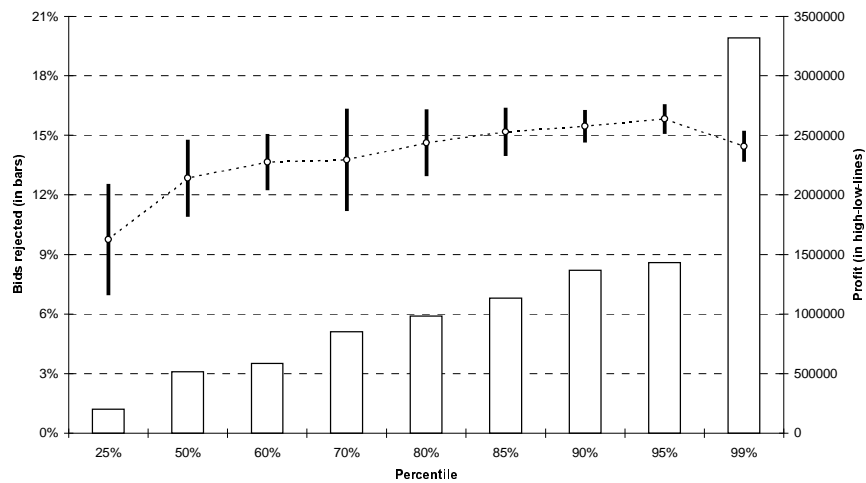


Figure 49: Sampling of percentile for the Lead-Neg/C policy applied to the simple two-tier model. Setting the percentile to 95 percent gives the optimal safety leadtime with respect to profit.

Additional experimental results for the experiment of Section 6.6.2

The performance metrics reported in Table 40 is obtained after fine-tuning the value of s_{max} to maximize the average profit for the 20 randomly generated experiments with the given resource efficiency.

| Eff | s_{max} | Bids | R% | Compl | Tardy | U% | Lead | Adj | INV | TAR | Revenue | Profit |
|------|-----------|------------|--------------|------------|------------|--------------|------------|----------|--------------|-------------|--------------|--------------|
| 1.00 | 1000 | 804 ±13 | 10.9 ±1.0 | 715 ±16 | 73 ±19 | 81.6 ±3.9 | 106 ±6 | 38 ±3 | 1149 ±86 | 111 ±58 | 6026 ±129 | 4766 ±199 |
| 0.91 | 1800 | 804 ±13 | 16.0 ±1.4 | 676 ±17 | 109 ±23 | 82.8 ±3.7 | 134 ±8 | 51 ±3 | 1356 ±89 | 220 ±73 | 5657 ±144 | 4080 ±229 |
| 0.83 | 2500 | 804 ±13 | 20.6 ±1.6 | 639 ±17 | 117 ±27 | 82.3 ±4.1 | 161 ±11 | 60 ±4 | 1498 ±96 | 310 ±119 | 5314 ±149 | 3506 ±296 |
| 0.71 | 4500 | 804 ±13 | 32.5 ±1.7 | 541 ±16 | 57 ±18 | 79.1 ±3.9 | 192 ±13 | 81 ±3 | 1711 ±106 | 169 ±88 | 4418 ±111 | 2537 ±228 |

Table 40: Performance metrics of the Sync-Neg/C policy when allowing for individual fine-tuning of the maximum buffer size s_{max} . The column denoted “Eff” gives the mean resource efficiencies.

Appendix C: List of symbols

| Symbol | Page | Symbol | Page | Symbol | Page |
|------------------------|------|--------------------|------|--------------------|------|
| α_B | 86 | l_{max} | 68 | \bar{q}_{bid} | 88 |
| α_R | 86 | l_{min} | 68 | q_{bid}^{max} | 84 |
| $\hat{\beta}_0$ | 65 | \hat{l}_s | 68 | q_{bid}^{min} | 84 |
| $\hat{\beta}_1$ | 65 | M_i | 79 | R_i | 78 |
| $\hat{\mu}$ | 81 | mic | 70 | r_{op} | 84 |
| $\hat{\sigma}^2$ | 81 | mic_{prod} | 84 | $random$ | 86 |
| $\hat{\sigma}_\beta^2$ | 67 | mic_{prod}^{max} | 83 | S_c | 78 |
| $\hat{\sigma}_p$ | 67 | mic_{prod}^{min} | 83 | S_{ll} | 67 |
| b_{price} | 86 | mtc | 70 | S_o | 78 |
| b_{prod} | 86 | n_{bid} | 86 | S_{ql} | 65 |
| $beta$ | 85 | \bar{n}_{bid} | 84 | S_{qq} | 64 |
| \hat{C} | 70 | n_{BOM}^{max} | 83 | t_{busy} | 86 |
| C_i | 79 | n_{BOM}^{min} | 83 | \bar{t}_{busy} | 84 |
| $C_{i,j}^{fixed}$ | 79 | n_H | 89 | t_C | 85 |
| $C_{i,j}^{inventory}$ | 79 | \hat{n}_i^{op} | 88 | \hat{t}_{comp} | 64 |
| $C_{i,j}^{tardiness}$ | 79 | n_{op} | 84 | t_{diff} | 73 |
| c_{bid} | 73 | n_{op}^{max} | 83 | t_i^C | 79 |
| c_{rej} | 73 | n_{op}^{min} | 83 | $\hat{t}_{i,k}^C$ | 80 |
| \hat{d}_{lead} | 70 | n_{prod} | 83 | t_i^D | 79 |
| d_{op} | 86 | n_{res} | 83 | t_n | 82 |
| d_p | 88 | P | 78 | $t_{profitable}$ | 73 |
| d_{range} | 84 | P_α | 67 | \bar{t}_R | 84 |
| d_w | 89 | p_{agent} | 84 | t_R | 86 |
| $E_{i,k}$ | 80 | p_{bid}^{min} | 84 | t_{repair} | 86 |
| e | 88 | p_{bid}^{max} | 84 | \bar{t}_{repair} | 84 |
| f_i^{bid} | 88 | p_{diff} | 73 | t_{req} | 64 |
| $gamma$ | 86 | $poisson$ | 84 | u_p | 84 |
| I_α | 82 | q_{bid} | 86 | \bar{u}_p | 88 |
| \bar{l} | 64 | q_{bid}^{max} | 84 | u^{max} | 83 |
| \hat{l} | 65 | q_{bid}^{min} | 84 | u^{min} | 83 |
| \hat{l}_i | 64 | \bar{q} | 64 | $uniform$ | 86 |
| l_i^N | 88 | q_{bid} | 86 | $Var(\hat{\mu})$ | 82 |