An Investigation of an Agent-Based Scheduling in Decentralised Manufacturing Control

Lawrence Ong

Selwyn College

Department of Engineering

University of Cambridge

A thesis submitted to the University of Cambridge
in partial fulfillment of the requirements
for the Degree of M.Phil. in Engineering

31 August 2003

DECLARATION

The work presented in this thesis is original and is my own work. To the best of my knowledge and belief, this thesis contains no material previously published by any other person except where due acknowledgment has been made.

Lawrence Ong
31 August 2003

ACKNOWLEDGEMENTS

This thesis is the result of a one-year research with Cambridge Auto-ID Centre in University of Cambridge, whereby I have been supported by many people. I would like to take this opportunity to express my greatest gratitude to all of them.

This thesis would not have been possible without the help of my supervisor Dr. Duncan McFarlane, who provided a motivating, enthusiastic, and critical atmosphere during many discussions we had. It was a great pleasure to carry out my research under his supervision.

I am highly indebted to my advisor, Dr. James Brusey, who provided constructive comments throughout this research period and helped me tremendously in software simulations.

I would like to express my gratitude to Chien Yaw, who gave me stimulating suggestions and encouragement from the commencement to the completion of this research.

Last but not least, I would like to thank my parents and my sister who never cease to support me in every possible way.

ABSTRACT

This thesis investigates a scheduling algorithm for a heterarchical, distributed manufacturing system. Scheduling in a heterarchical, distributed system is a challenge as the algorithm must be able to operate in the absence of a coordinator and global information. A distributed scheduling method using a market based scheduling algorithm is investigated. Not only is this algorithm able to operate in the absence of a coordinator and global information, it also provides a good match between the software processing parts and the actual physical entities in a manufacturing system. In addition, each entity in the system is able to retain its private information.

It is found, using simulations, that the market based scheduling algorithm is able to give close-to-optimum solutions. There are instances when the algorithm does not perform well. The lower bound of the algorithm's performance is derived and conditions under which the algorithm does not perform well are investigated.

The market based scheduling algorithm is successfully implemented in the Cambridge Auto-ID Lab, an HMS testbed. Modifications are made to the original algorithm so that it can cope with the dynamics of the real time system and handle rush orders. It is shown in simulations and in a physical system that the algorithm is able to perform real time rescheduling, handle rush orders and produce good scheduling solutions according to some performance measures.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE AND LIST OF SYMBOLS

HMS	Holonic manufacturing system
MAS	Multi-agent system
CDA	Continuous double auctions
SFB	Straight forward bidding
SFBSCA	Straight forward bidding with sunk cost awareness
MBSA	Market based scheduling algorithm
$\mathbf{G}=\{G_0, G_1, \ldots, G_{N-1}\}$	A set of N discrete resources
N	Total number of resources
q_j	The reserved price of resource G_j
$q_{\max}$	The highest value of all N reserved prices
$q_{\min}$	The lowest value of all N reserved prices
p_j	The current price of resource G_j
M	Total number of bidders
$\mathbf{X}_i$	A set of resources that bidder i is holding. $\mathbf{X}_i \subseteq \mathbf{G}$
$v_i(\mathbf{X}_i)$	The value that bidder i will get when it is holding $\mathbf{X}_i$
v_i	The value of bidder i if it can get all required resources
$\mathbf{f}=[f_0 \ f_1 \ \ldots \ f_i \ \ldots \ f_{N-1}]$	A solution of a scheduling problem
$\mathbf{U}=\{j \mid f_j=-1\}$	Set of indices of unallocated resources
ε	Bid increment after each round of bidding
S_i	A slot, which is a discrete resource in job shop scheduling
$T_{travelling}$	Maximum time for a product to travel from its storage area to the vicinity of a machine
H_i	Maximum surplus of a bidder
$T_{operation}(K)$	Operation time for a product
T_{main_loop}	Time for a shuttle to travel once round the main loop
$T_{side\_loop}(K)$	Time for a shuttle to travel once round a side loop

Q	Maximum number of product at the vicinity of a machine
$\Re^{+}$	The set of all positive real numbers
Z^{+}	The set of all positive integers
$\lceil \bullet \rceil$	Ceiling function
$\lfloor \bullet \rfloor$	Floor function

CHAPTER 1 INTRODUCTION

1.1 Overview

This research investigates a scheduling algorithm for a heterarchical, distributed manufacturing system. In a manufacturing system with resources and jobs to be serviced by these resources, scheduling is to allocate the resources to the jobs. Scheduling in a heterarchical, distributed system is a challenge as the algorithm must be able to operate in the absence of a coordinator and global information. In this thesis, a market based scheduling algorithm (MBSA) based on the work by Wellman [6] is investigated. Not only is this algorithm able to operate in the absence of a coordinator and global information, it also provides a good match between the software processing parts and the actual physical entities in a manufacturing system. In addition, each entity in the system is able to retain its private information.

In the first stage of this research, the performance the MBSA is studied. It is found that the MBSA is able to give close-to-optimum solutions. A bound of the algorithm's performance and conditions under which the algorithm does not perform well are derived. These analyses help manufacturing factories in designing the scheduling parameters when using the MBSA.

In the second stage of this research, the implementation issues of the MBSA in a holonic manufacturing system (HMS) ([17]) are studied. Modifications to the algorithm are proposed so that it can operate in a dynamic system and handle rush orders. It is shown from simulations and implementation of the algorithm in the Cambridge Auto-ID Lab that the MBSA is able to meet these requirements.

1.2 Research Objectives

Parunak pointed out that any successful application of a technology must reconcile two perspectives, which are to focus on a particular capability and to demonstrate how this capability can be used in practical problems [32]. In this research, both the theoretical aspects and the practical aspects of the MBSA are investigated. The objectives of this proposed research are:

1. To investigate the behaviour of the MBSA.
2. To study the implementation issues of the MBSA in a real time HMS.

With these objectives, this research sought to answer the following questions:

1. What is the performance of the MBSA under various scheduling conditions?
2. What are the instances when the MBSA performs badly and the reasons it does not perform well?
3. What is the bound of the performance of the MBSA?
4. What are the issues or difficulties in implementing the MBSA in a real time HMS?
5. What modifications are required for the MBSA to operate in a real time HMS?
6. Is the MBSA able to operate in a real time HMS after these modifications?

1.3 Research Methodology

A methodology is developed to achieve the objectives and to answer the research questions. This research is divided into two phases:

1.3.1 Software Simulations and Mathematical Analysis

The first phase attempts to achieve the first objective and to answer the first three research questions. In this phase, the following are done:

1. The MBSA is simulated under various scheduling conditions and its performance is studied.
2. The bound of the performance of the MBSA is derived.
3. Conditions under which the MBSA does not perform well are investigated.

1.3.2 Implementation of the Market Based Scheduling Algorithm in a Real Time Holonic Manufacturing System

The second phase of the research attempts to achieve the second objective and to answer the last three research questions. The implementation issues of the MBSA in a real time HMS are investigated. In the literatures, Kumar discussed an application for auctioning over the Internet [30]. University of Michigan designed an auction server, Michigan Internet AuctionBot, for testing electronics auctions [31]. Both of these works highlighted a few issues (for example, information delay, interactions between the auctioneers and the bidders) in implementing auctions in distributed, electronic systems. Attentions are paid to these issues when implementing the MSBA.

1.3.2.1 Implementation of the Market Based Scheduling Algorithm in a Simulated Lab Environment

Firstly, the MBSA is implemented in a simulated lab environment. This stage is included prior to the implementation of the algorithm in a physical system as it is anticipated that many issues might arise in the implementation stage. The reasons for simulations are:

1. It is easier to manage a simulation than a physical system. It takes less time to reset the experiment setup for repetitive experiments.
2. The speed of objects' movement and manufacturing processes can be scaled easily. This speeds up the experiment.
3. Hardware issues and software issues (the scheduling software program) can be easily separated. This makes debugging of the scheduling program easier.

1.3.2.2 Implementation of the Market Based Scheduling Algorithm in Cambridge Auto-ID Lab

Secondly, the MBSA is implemented in Cambridge Auto-ID Lab, which implements an HMS. The aim in this stage is to investigate implementation issues which are not encountered in the simulations and to verify the results obtain in simulations.

1.4 Organisation of this Thesis

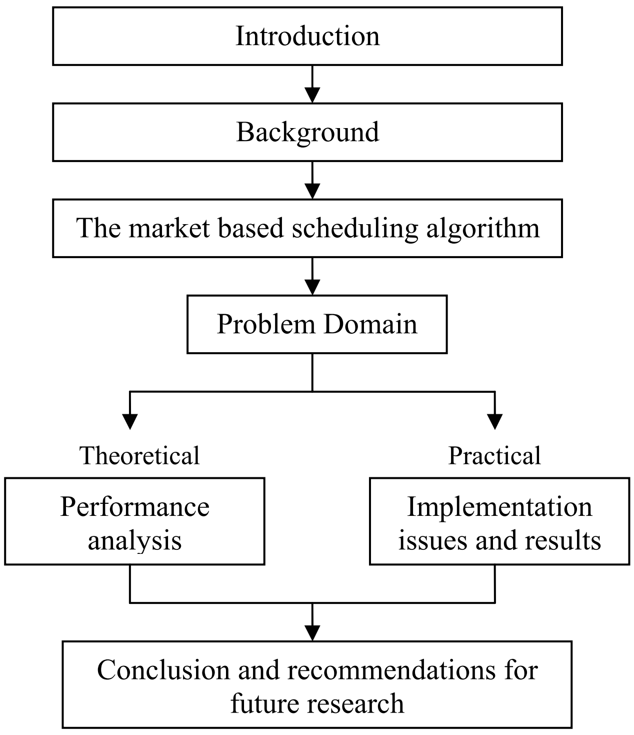

Introduction
Background
The market based scheduling algorithm
Problem Domain
Theoretical
Practical
Performance analysis
Implementation issues and results
Conclusion and recommendations for future research

Figure 1. Organisation of this thesis.

This thesis is organised as depicted in Figure 1. Chapter 2 gives brief introductions to multi-agent systems (MASs), HMSs, scheduling and auctions. Chapter 3 gives a detail description of the MBSA. As the algorithm uses an auction as the scheduling mechanism, different bidding strategies could be employed by the bidders. Two bidding strategies used in this research are introduced at the end of this chapter. Chapter 4 lays down the problem domain of this research. The type of scheduling considered in this research is presented. Chapter 5 and Chapter 6 scrutinise the MBSA by using simulations and analyses. The former chapter looks at how different scheduling conditions affect the performance of the algorithm. The latter chapter presents the analysis of the bound of the algorithm's performance and derivations of the conditions under which the algorithm produces suboptimum solutions. Chapter 7 presents the results of implementing of the MBSA in a simulated environment and a real time HMS. A few issues of implementing the algorithm in a real time, dynamic manufacturing system are highlighted. The final chapter presents our original contributions in this research, reiterates important findings and gives recommendations for future research.

1.5 Contributions

Our original contributions in this research are:

1. We compared the performance of the MBSA under various scheduling conditions.
2. We derived the bound on the performance of the MBSA.
3. We investigated conditions under which the MBSA always produces optimum solutions.
4. We proposed modifications to the MBSA for it to operate in a dynamic manufacturing system.
5. We successfully implemented the MBSA in a HMS.

CHAPTER 2 BACKGROUND

This chapter gives a brief introduction to MASs, HMSs, scheduling and auctions. We describe the relationship and the differences between an MAS and an HMS. We then discuss the characteristics of a heterarchical HMS and the challenge faced by scheduling in a heterarchical HMS. Various scheduling techniques are then surveyed and their pros and cons are presented. This chapter wraps up with a brief introduction to auctions, which form the basis mechanism of the MBSA.

2.1 Multi-Agent Systems

Since 1980s, MASs have grown into one of the most active areas of research and development in computing [25]. In an MAS, agents are situated in an environment and they are able to “see” the environment through sensors and can possibly change the environment through their actions. The agents are autonomous that they are capable of making decisions based on their knowledge about the environment and/or other agents without the intervention of other agents. The agents exhibit social behaviour, whereby they interact with other agents in order to achieve their goals. This section discusses three different types of agent architectures. The advantages and disadvantages of each type of agent are summarised. This is followed by looking at how, in an MAS, the agents can reach agreements. Particular attention is paid to the auction-based negotiation among agents.

2.1.1 Types of Agents

Wooldridge [13] categorised agents into three categories: deductive reasoning agents, practical reasoning agents and reactive agents. Deductive reasoning agents and practical reasoning agents use symbolic representations to represent the environment, the agents' beliefs, desires and intentions. In contrast, the reactive agents do not use symbolic representations.

A deductive reasoning agent gets information about the environment through its sensors and translates it to symbolic descriptions, which define its *state*. It scans through a set of deductive logical rules (For example " $\mathrm{IF}\,\alpha_1\ \mathrm{AND}\,\alpha_2\ldots\mathrm{THEN\,DO}\,\zeta$ ", where $\{\alpha_i, i = 1,2,\ldots\}$ are symbolic descriptions and ζ is an action) and select an action in a deductive rule which the antecedents (α_i) match its state. One disadvantage of the deductive reasoning approach is that the environment may change when the agent scans through the logical deductive rules for an appropriate action as well as during the action itself. Also, the search for an action may take a long time.

Instead of following deductive rules, a practical reasoning agent determines what it wants to do (deliberation) and decides how to do it (means-ends reasoning). From the agent's beliefs about the environment (again, updated through its sensor) and previous intentions, it generates a set of intentions. It then works out a plan (a sequence of actions) to achieve its intentions. As the search over all possible sequence of actions may be computationally extensive, a plan library is usually available. The library consists of pre-computed plans which map the current states of an agent to the states after a sequence of actions.

Both deductive reasoning agent and practical reasoning agent use symbolic representations of the physical environment, the agent's beliefs, intentions and desires. However, Wooldridge pointed out that it may not be obvious how the mapping between the physical environment and the symbolic representation may be realised [13]. A reactive agent does not use any complex symbolic representation or symbolic reasoning. It simply maps what it senses to an action. The architecture of reactive agents is simple. But, the disadvantages are that the decision making process is made using local information and that it may be difficult to build reactive agents such that a desired overall behaviour can be guaranteed by the interactions among individual agents and the environment.

For simple bidding strategies and bid evaluations in the MBSA, an MAS with reactive agents is a good choice as its architecture is simple and there is no complex symbolic representation required in a bidding system. Furthermore, with the lack of a coordinator and global information in a heterarchical, distributed system, an agent only uses local information to determine its actions.

2.1.2 Agent Negotiations

After describing different types of agents, we now look at how agents interact with one another. Kraus [26] presented six techniques for reaching agreements in an MAS: strategic negotiation, auction, coalition formation, market-oriented programming, contracting and logical argumentation. We briefly describe these negotiation methods and their suitability to be implemented in a manufacturing environment, particularly in scheduling.

In the strategic negotiation model, every agent takes turn to make an offer (a possible agreement). If all agents accept the offer, the negotiation terminates and the offer is implemented. A conflict results if one or more agents opt out and the negotiation ends. Otherwise, if not all agents agree on that offer but no agent opts out, the negotiation continues with the next agent proposing a counteroffer. The strategy of each agent must be designed such that its offer is better for any agent than to opt out. The strategic negotiation model provides a solution to a wide range of problems [26]. Jennings [27] pointed out a few problems in the strategic negotiation model. In a multi-issue case, it may be difficult to define the agent's preferences over outcomes. The search for mutually acceptable solutions may be computationally time consuming. Furthermore, in order to find these solutions, an agent needs information of other agents. This may not be possible when the agents are competing.

McAfee [15] defined an auction as "a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants". One attraction of using auction to reach agreements is its simplicity in the interactions among agents. Auctions are described in greater detail in Section 2.4 on page 23.

Market-oriented programming is an approach based on market price mechanism [29]. In WALRAS, a market-oriented programming environment developed by Wellman [28], there are two types of agents. The consumer agent can buy, sell or consume goods. The producer agent transforms one type of goods into another. Each agent gives its quantity (demand or supply) vs. price curve to an auctioneer. The auctioneer determines the market price for each good. The competitive equilibrium is reached when the excess demand in the market approaches zero. However Wellman pointed out that system equilibrium or optimum solution may not be achieved without a central control. Auction can be classified as a special case of market-oriented programming.

Cooperation can be achieved by forming coalitions. An agent will join a coalition if it gains more from joining than not. This is useful in distributing jobs among agents. But in a manufacturing environment where agents are fighting for resources, it begs the question of how the resources can be allocated. Other negotiation techniques will have to be employed to solve the problem.

In contracting, an agent convinces other agents to help it with its job by giving them rewards. This method is useful in job decomposition. Contracting is often used in a factory operation cell, where machines contract out their operations to individual tools, for example drilling, clamping, and milling. After a factory schedule is made, each machine can use this negotiation technique to communicate with its sub-parts to perform a job.

Logical argumentation is usually used in symbolic-based agent architecture. In this model, an agent tries to influence the intentions of other agents. During the negotiations among agents, each agent receives messages from other agents and updates its beliefs and intentions. This can be employed in an auction if the agents have information about other agents. When an agent knows that both itself and another agent can be better off by exchanging their resources, it can persuade the agent to do so.

In the next section, we will discuss HMSs, which have useful parallels to MASs.

2.2 Holonic Manufacturing Systems

2.2.1 What is a Holonic System?

The concept of holonic system was proposed by Arthur Koestler. The word “holon” comes from the Greek word “holos”, which means whole and the suffix “on” which suggests a particle. He observed that living organisms and social organisations are made up of sub-wholes or parts which are themselves self-contained wholes to their sub-parts. The Holonic Manufacturing Systems Consortium translated the concept of holons to the concept for the manufacturing arena. The goal is to achieve the benefits that a holonic system provides to living organisms and societies to the manufacturing industries [34].

In an HMS, a holarchy refers to a system of holons interacting in a cooperative manner to achieve certain goals. Holons are autonomous and cooperative building blocks in the HMS. A holon consists of an information component and often also a physical component. Examples of the physical components are products, and machines. The information processing part makes decisions, manages the physical components and interacts with other holons.

Chirn and McFarlane surveyed a few proposed system architectures for the HMS [17]. One of them that received much attention is MAS. This is due to the fact that an agent is autonomous and proactive, which is consistent with the concept of a holon that is autonomous and cooperative. The cooperative nature of the agents can be realised by proper design of rules that govern the agents. The difference between HMS and MAS is that the former is *an organizing principle for structuring and controlling manufacturing*; while the latter is *a software technology that realizes the information processing of the HMS* [39].

2.2.2 Why a Holonic Manufacturing System?

An HMS has the following characteristics which make it an attractive solution for manufacturing companies (some of them are taken from [40])

1. A distributed system is possible where decisions are made locally. A distributed system is more tolerant to single point failures.
2. Plug-n-play capability. New components can be added easily to the existing system.
3. An HMS is resistant to disturbance. This is one advantage of a distributed system over a centralised system.
4. An HMS is re-configurable and flexible. In a conventional manufacturing approach, the system is often configured and optimised for a specific product type. Whenever a product needs modification, re-configuration of a system requires extensive effort. In an HMS, a machine represents a capability and is not configured for a fixed product type. Any change in the product design only requires different interactions between the product holons and the machine holons.
5. Complex structures can be configured via aggregation of lower level unit operations. This simplifies the system design, trouble shooting, future modifications and upgrades.

2.2.3 Hierarchical vs. Heterarchical System

In this section, we discuss the differences between a hierarchical system and a heterarchical system. Attentions are paid to the types of communications among agents and the use of global information.

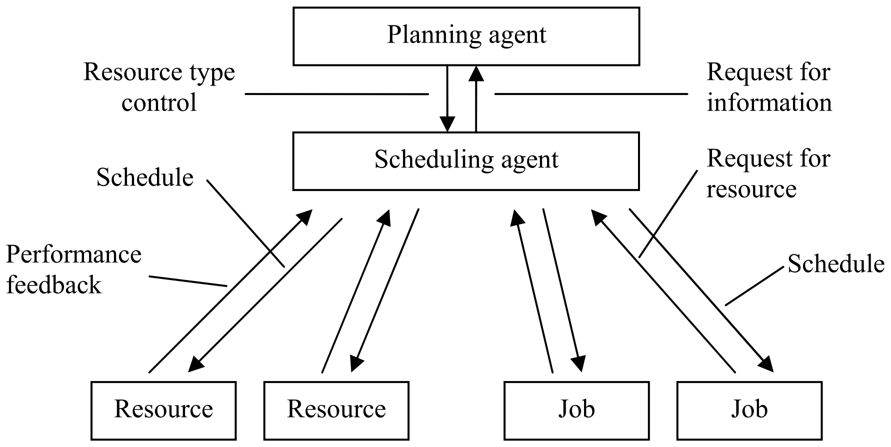


Figure 2. An example of a hierarchical system.

In a hierarchical system, each agent represents a function capability [33]. There exists a master-slave relationship among the agents. A master agent requests a slave agent to perform certain jobs and expects specific results. An example is given in Figure 2. The jobs send their request to the schedule agent while the schedule agent requests information from the planning agent. The planning agent decomposes jobs into sequence of operations and determines the types of resource required for each operation. It has the information of all resource types in the system. The scheduling agent produces a schedule that defines timing specifications for each operation. It holds the schedule of the entire system.

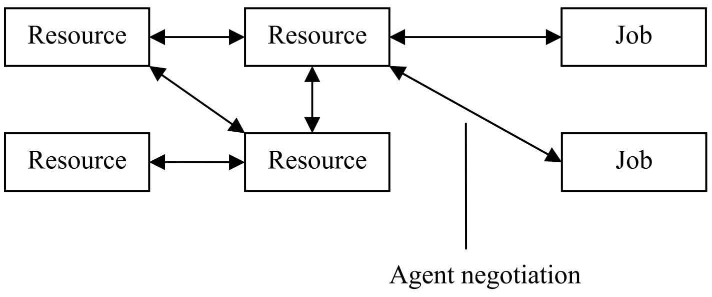


Figure 3. An example of a heterarchical system.

On the other hand, in a heterarchical architecture, agents communicate as peers. There is no master-slave relationship. Though there are rules defining the agent negotiations, one agent does not have the power to dictate another agent. No global information is kept anywhere. An example of a heterarchical system is depicted in Figure 3. Here jobs negotiate with resources to use their service; a resource may request the service of another resource. Each job and each resource keeps its own schedule and there is no agent that holds the overall schedule.

The heterarchical architecture is inherently capable of self-configuration, scalability, fault tolerance and emergent behaviour [33]. In the hierarchical architecture, the agents in the lower layer may not work properly if the agents at the higher level fail. In a heterarchical system, if some of the agents are destroyed, the remaining agents continue to function, since control is distributed. A coordinated activity and decision (for example, how much to produce) emerges from many local decisions (for example, the schedule at individual machines). McEleney argues that the distributed approach lowers the complexity of a system by concurrent local decision making, avoiding a single but complex coordination of the whole system [36].

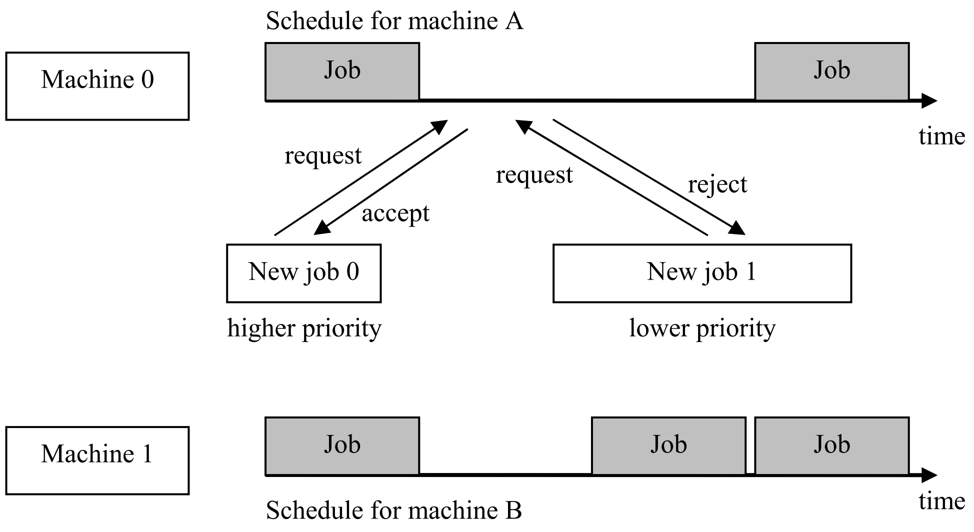


Figure 4. A scheduling problem with two resources and four jobs.

However, the heterarchical system lacks higher level agents that oversee the entire system. As a result, higher level coordination may not be trivial. The lack of central control and global information poses a great challenge to scheduling in a heterarchical system. An example is given in Figure 4 where two new jobs arrive and both try to get a processing slot from machine 0. Machine 0 accepts the request from new job 0, which has higher priority. We see that in this case, new job 1 cannot be scheduled. As opposed to a heterarchical system, in a hierarchical system where both the schedules for machine 0 and machine 1 are available to a higher level (scheduling) agent, it can schedule new job 0 to machine 1 and new job 1 to machine 0, so that both jobs can be processed.

2.2.4 Scheduling Requirements in a Heterarchical Holonic System

Due to the lack of central control, a heterarchical system needs a scheduling algorithm with the following characteristics. In the following list, 1 to 3 are requirements, while 4 to 7 are desirable properties.

1. The algorithm must be distributed. In a heterarchical system, there is no single agent that performs scheduling. Scheduling is carried out over a number of processors that are able to communicate with each other. However, to date, most holonic scheduling approaches are centralised to some degree [18]. Usually a coordinator is required to guarantee certain performance level of the whole system.
2. The algorithm must be able to cope with the absence of global information.
3. The algorithm must be able to operate in a dynamic system. This characteristic is not just pertaining to heterarchical system but to manufacturing systems in general. In a manufacturing environment, jobs arrive randomly. The algorithm must be able to update its schedule when new jobs arrive.
4. The algorithm is scalable. In a distributed system, information processing is done at various distributed processors. It is desirable that as the system gets larger (for example, when the number of machines or the number of jobs increases), the information processing load is shared evenly among processors.
5. The algorithm can handle *rush* order. A rush order is an order that has high priority and comes late.
6. Each information processing unit in the distributed scheduling mechanism has their corresponding physical entities. This is desirable as the processing load is automatically distributed when a new job arrives or a new resource is installed. In addition, physical correspondence is consistent with the concept of holonic system.
7. Each job and resource can retain its private information. This is a desirable property, for example when the jobs are from different companies that are self-interested and competing among each other, they may not want to reveal their private information to other companies.

2.3 Scheduling

2.3.1 What is Scheduling?

The theory of scheduling spans several disciplines from the industrial research to the academic research in Operational Research, Manufacturing, Computer Science, Electrical Engineering, Communications, and Applied Mathematics. Scheduling is the process of allocating a set of resources to different jobs. In the context of this thesis, only job shop scheduling is considered. Johnson defines job shop scheduling as "… determining the order or sequence in which the machines will process the jobs so as to optimise some measures of performance." [10].

2.3.2 Distributed Techniques vs. Distributed Problems

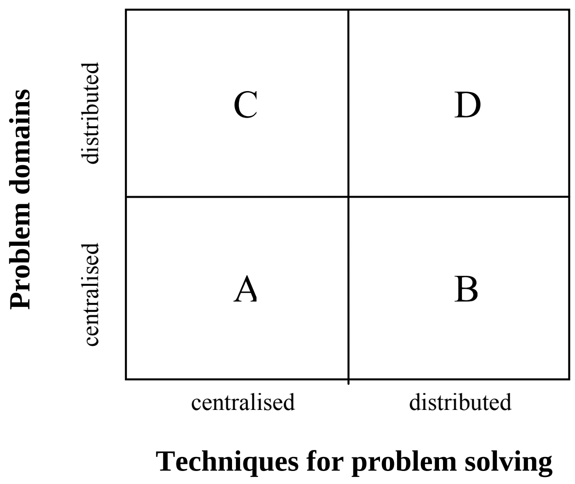


Figure 5. A matrix showing centralised/distributed techniques for problem solving and centralised/distributed problems.

Ferber made a clear distinction between a distributed technique for problem solving and a distributed problem [2]. A distributed technique for problem solving refers to the case where a problem is solved by using different processors, each performing some computations and communicating with each other (refer to sectors B and D in Figure 5). Independent of the technique used, the problem domain may be distributed or centralised. A distributed problem refers to a system which is made up of entities interacting with each other while each of them requires its own processing (refer to sectors C and D in Figure 5). For example monitoring of energy or telecommunications network where each node in the network is physically distributed. Each node needs local supervision. In an HMS where the manufacturing entities, for example machines, products, and orders, are autonomous, the scheduling problem domain is also distributed.

2.3.3 Currently Used Scheduling Algorithms

Traditionally, research into scheduling focused on minimising the schedule length or the mean flow time [9]. Mean flow time is the average time taken from the start of the schedule to the time a job is completed. The techniques used are centralised. However, centralised manufacturing scheduling and control mechanisms are found to be vulnerable to single point failures [5]. In addition, traditional scheduling techniques assume a static environment, whereby all necessary information is known beforehand [4]. Real time scheduling in manufacturing is inherently dynamic as orders arrive at different times. Also, disturbances such as machine breakdowns may happen. Thus a scheduling algorithm that is able to handle a dynamic environment and disturbances is desired.

While centralised algorithm is a sensible one to be used in a centralised domain, many distributed systems also use centralised algorithms. This is because centralised scheduling has being well studied and there are many algorithms that can be applied for different system requirements. Bussmann [4] proposed a multi-agent approach using a centralised scheduling technique for HMS, whereby the schedules of transporters (machines that move products from a point to another) are determined by a coordinator.

When new orders arrive, the coordinator announces jobs to the transporters. The transporters send their capabilities to the coordinator, telling if they are able to handle the jobs. The coordinator searches for an assignment that can cover the maximum number of jobs. Here a centralised scheduling technique is used where the search is done solely by the coordinator. Bussmann pointed out that a totally distributed approach is not appropriate. He claims that a central component is required to globally optimise a schedule.

Kis [37] used a centralised scheduling algorithm in a distributed manufacturing system employing the market based mechanism. The first price sealed bid auction is used. Here a management agent is used to coordinate the schedule. It received orders from different companies and accepts a number of them. It then announces new jobs to machines and the machines submit bids to the management agent. The management agent produces a schedule by assigning jobs to machines with the highest bids.

Lin and Solberg [38] proposed a distributed scheduling algorithm for a distributed system, that uses a bidding mechanism. In their model, orders that arrive at the system announce their jobs to the machines. The machines submit bids to the orders; the orders evaluate and select machines with the best bids. This distributed approach is scalable and it has a good mapping between the information processing parts and the physical entities. However, it is not clear how high priority order differentiates itself from the others. Also, a job is committed when an order accepts the best bid from a machine.

Gozzi [19] proposed a distributed scheduling technique via agent negotiations using a heuristic approach. In the proposed architecture, every job calculates its best starting time to be processed and sends its proposal to various machines with a bid price. The machines either reject, accept or offer counter proposals to the jobs. However, this scheduling technique does not assign priorities to the jobs. Also, its ability to cope with rush orders is not documented.

Gou [35] investigated a distributed scheduling algorithm in an HMS using Lagrangian relaxation method. It decomposes the scheduling problem into sub-problems. Each sub-problem finds the schedule for one job. A schedule is produced after many iterations. In each iteration, a central coordinator works out the Lagrangian multipliers for each sub-problem. With these multipliers, the machines find the best starting time for each job by solving the sub-problems. With the new updated job starting times, the central coordinator works out new Lagrangian multipliers. This process is repeated until the algorithm converges after a certain number of iterations. The advantage of using the Lagrangian relaxation method is that it guarantees a near-optimum solution [33]. Also, the algorithm can be mapped into an HMS. However, this algorithm does not prioritise jobs. It may suffer scalability problem because the machines are responsible for solving the sub-problems. So when the number of jobs increases, the machines will be loaded.

Other researches on scheduling in HMSs include McEleney [36] who investigated a centralised algorithm and Xiao Qin who looked at a dynamic, fault-tolerant scheduling for heterogeneous distributed systems ([23], [24]). Many researches in distributed manufacturing scheduling have looked into the system architectures, in which auctions were briefly mentioned to be used for scheduling and planning ([4], [7]).

2.3.4 Advantages and Disadvantages of the Distributed Scheduling Algorithm

The centralised scheduling technique has a few limitations:

1. The centralised scheduling technique assumes that all information required to carry out the scheduling is available at a processor in which calculations are performed [1].
2. The centralised scheduling technique is not desirable in a heterarchical system where it is difficult to obtain a totally centralised overall view. As the system gets larger, controlling and monitoring of individual nodes gets more complicated.
3. In a distributed environment without central control, it is not easy to determine which processor is responsible for the information processing for scheduling.
4. The centralised technique in a distributed system assumes distributed information with cooperative behaviour [1]. A centralised approach may be difficult if individual parts are not cooperating.

Besides overcoming the limitations faced by the centralised approach, the distributed approach has the following advantages:

1. It is more economical to process information in a large number of central processing units (CPUs) than one single mainframe computer. The computing power of a CPU is proportional to the square root of its price. This is known as the Grosch's law [11].
2. The distributed technique is more reliable. If one computer crashes, scheduling may still continue if information processing runs on separate processors [12].

However, the distributed approach has a few disadvantages:

1. Extra processing load is incurred as message passing is required among distributed nodes for coordination.
2. Performance of the system may be affected by the communication links between nodes. For example, the throughput of an auction deteriorates quickly when the number of parts increases to about fifty [20].
3. While the distributed approach is designed to cope with disturbances, there is, in general, a trade-off between its performance and the reactivity of the system to disturbances [3].
4. Myopic decision may occur due to the lack of global information [38].

2.4 Auctions

In this section, we discuss different types of auctions and different bidding strategies that the bidders can employ in an auction. Auction is the basis of the MSBA, which is the scheduling algorithm that we investigate in this thesis.

2.4.1 Types of Auctions

Four basic types of auctions are: the English auction, the Dutch auction, the first-price sealed-bid auction, and the second-price sealed-bid (or Vickrey) auction [15].

In the English auction, the price of goods is successively raised until one bidder remains. In the Dutch auction, a high initial price is called. The price is lowered until a bidder accepts the price. In the first-price sealed-bid auction and the second-price sealed-bid auction, the bidders submit their bids and the highest bidder gets the goods. The difference between the first-price seal-bid and the second-price sealed-bid is that in the former the winner pays the price it bids but in the latter the winner pays the price of the second highest bid.

Table 1 gives a summary of these four types of auction. In an open-cry auction, each bidder can see at what price other bidders bid. In the closed-cry auction, the bidders submit their bids to the auctioneer. A bidder does not know the bidding prices of other bidders.

Table 1. Comparison between different auctions.

Auction	Auction type	Open-cry / closed-cry?	How does the price change?	What price does the winner pay?
English auction	Ascending-bid	Open-cry	Increases in each round of bidding	Bid price of the highest bidder
Dutch auction	Descending bid	Open-cry	Decreases in each round	Bid price of the highest bidder
First-price sealed-bid auction	Sealed bid	Closed-cry	Auction closes after the first round of bidding	Bid price of the highest bidder
Second-price sealed-bid auction	Sealed bid	Closed-cry	Auction closes after the first round of bidding	Bid price of the second highest bidder

Other auctions variants include continuous double auctions (CDA) [42] and combinatorial auctions [41]. CDA are useful when similar and substitutable goods are available in a large number. However, in this thesis, we consider discrete goods that are not substitutable. A scheduling system that uses combinatorial auctions is computationally complex [6] and it is often not practical because of the difficulties in coordinating the allocation of resources [1].

2.4.2 Bidding Strategies

In an English auction where the amount of money that each bidder has and its evaluation of the goods are kept private, the bidder's dominant strategy is to bid a small amount higher than the current highest bid until the price exceeds its willingness to pay ([8], [15]). However, this strategy does not work well in CDA and combinatorial auctions.

In CDA, a seller needs to acquire information about other sellers and the buyers before deciding on the ask price. On the other hand, a buyer needs to know the ask price and the bid price for a particular goods in previous rounds of auctions before submitting its bid [21]. For combinatorial auctions, a buyer needs to work out bid values for different combinations of items, where some items can compliment or substitute each other.

Anthony and Jennings surveyed various bidding strategies used in different auctions [22]:

1. In the possibility-based approach, a bidder evaluates the possibility of it winning the round for a given bid price. It then selects the most preferred decision according to a certain global utility value. This requires prior information, such as the winning prices in previous auctions and the money other agents may have.
2. In the recursive modelling method, a bidder models what other bidders do. The process can be recursive when a bidder models other agent's model. The modelling ends when the bidder has no deeper knowledge.
3. Using a negotiation decision function, bidders and sellers negotiate with offers and counter offers.
4. Using the stochastic modelling, a seller or a bidder models the auction process using a Markov Chain. Using the Markov Chain model, the seller or the bidder

is able to capture the variables that represent the dynamics of the auctions process, for example the number of sellers and buyers, the arrival rates of the buyers and the sellers, the distributions of the buying and selling prices.

(1), (2) and (4) require a bidder to monitor the bidding process of other bidders or to have the bidding information of other bidders in previous auctions. For (3), a bidder must know which bidder is holding which resources. A bidder will have difficulties using these strategies if it is new to the system and does not have information about other bidders. In this research, we look at the case where a bidder only considers the current highest bids for the resources and the money it has when it bids.

In this chapter, we have presented MASs and HMSs which form the architecture of the manufacturing system that this research concerns. We discussed various scheduling techniques found in the literature and their pros and cons. We then introduced auctions, which are used by the MBSA as the scheduling mechanism.

CHAPTER 3 A MARKET BASED SCHEDULING ALGORITHM

In this research, we investigate the market based scheduling algorithm (MBSA) in a distributed, heterarchical manufacturing environment. The MBSA is based on the model developed by Wellman [6]. This algorithm uses an English auction as the "platform" for scheduling. Section 3.1 lists the characteristics of the MBSA that makes it a suitable choice for a heterarchical, distributed manufacturing system. The rest of the chapter describes the scheduling model and the scheduling algorithm in detail. Two bidding strategies that a bidder can employ are presented.

3.1 Characteristics of the Market Based Scheduling Algorithm

This scheduling algorithm has the following characteristics, which satisfy the requirements for scheduling in a heterarchical system:

1. The algorithm uses a distributed technique and it can be implemented without the presence of a central coordinator.
2. No global information is required for the algorithm to work.
3. There is no restriction that the algorithm must be used in a static environment. We look at how the algorithm can be used in a dynamic environment.
4. This algorithm is scalable. The processing of bidding decisions is done by individual bidders. Hence, the computation load can be distributed when more bidders enter the system.
5. The original algorithm does not document the handling of rush orders. However, we adjusted the algorithm such that rush orders can be dealt with.

6. There is a mapping between the information processing parts of the algorithm and the physical entities in an HMS.
7. Each bidder can retain its private information, which is the maximum amount that it is willing to bid.

3.2 Scheduling Model

The scheduling problem is to allocate a set of discrete resources to a number of jobs. In an HMS that uses an MAS as the software architecture, each resource and each job constitute a holon and an agent is used to control the holon. In the MBSA proposed by Wellman, the job agents bid for the resources by submitting their bids to an auctioneer agent. For the rest of this thesis, we term the job agents as bidders.

We use the following notations in our analyses throughout this thesis.

$\mathbf{G} = \{G_0, G_1, ..., G_{N-1}\}$ is a set of discrete resources.

q_j is the reserved price of resource G_j.

$\mathbf{p} = \{p_0, p_1, ..., p_{N-1}\}$ is a set of prices where p_j is the current price of resource G_j. The price for a resource is the minimum amount that a bidder must bid to get the resource.

There are M bidders in the system, which we term as bidder 0, bidder 1, …, bidder $M-1$.

$\mathbf{X}_i$ is the set of resources that bidder i is holding. $\mathbf{X}_i \subseteq \mathbf{G}$.

$v_i(\mathbf{X}_i)$ is the value that bidder i will get when it is holding $\mathbf{X}_i$.

v_i is the value that bidder i will get when it holds all the resources that it needs.

$\mathbf{f}$ is a solution of a scheduling problem, which is an allocation of the resources to the bidders.

$\mathbf{f} = [f_0 \; f_1 \; \cdots f_i \cdots f_{N-1}]$ where $f_j \in \{-1, 0, 1, ..., M-1\}$ means that the *j*-th resource G_j is allocated to bidder f_j. $f_j = -1$ means that the *j*-th resource is unallocated.

$\mathbf{U} = \{j \mid f_j = -1\}$ is the set of indices of unallocated resources.

Each bidder needs a number of resources to complete its job. If it holds less than the number of required resources, the bidder's value is zero. Wellman defines the global value of a solution, $v(\mathbf{f})$ as the sum of the values of all bidders and the reserved prices of all unallocated resources [6]. This can be written as

$$v(\mathbf{f}) = \sum_{j \in \mathbf{U}} q_j + \sum_{i=0}^{M-1} v_i(\mathbf{X}_i) \qquad \text{Q 1}$$

3.3 Scheduling Algorithm

The scheduling algorithm follows the setting of an English auction. The auction mechanism is as follows:

1. Each resource is marked with a reserved price. The price of a resource that has not being bid for equals its reserved price.
2. If a bidder wishes to bid for a resource G_j, it must bid at a price not lower than the price p_j.
3. When the bid price from a bidder is higher or equal to the price of a resource, the resource is temporarily allocated to the bidder. The allocation here is temporary as other bidder may outbid the current highest bidder.

4. The price of a resource G_j is increased to $p_j = \beta_j + \varepsilon$ where β_j is the highest bid price up to that time and ε is the price increment for each bid. For the rest of this thesis, we set $\varepsilon = 1$.
5. Steps 2, 3 and 4 are repeated until no bidder bids further or after certain amount of time.

When the auction ends, a schedule is formed by allocating each resource to the highest bidder.

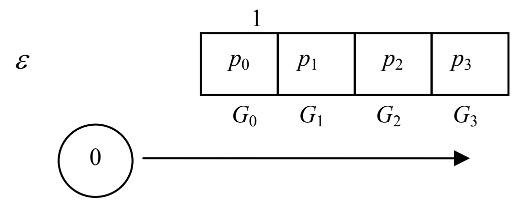


Figure 6. A bidding diagram.

Figure 6 depicts a bidding diagram used throughout this thesis. It shows the bidding process that occurs in an auction. ε denotes the price increment after each round of bidding. In the diagram, four discrete resources (indicated by the rectangles) are available for bidding, namely G_0, G_1, G_2, and G_3. At that instance, the prices for the resources are p_0, p_1, p_2, and p_3 respectively. A circle represents a bidder. The arrow represents the bid that a bidder places. In the example above, bidder 0 bids for G_3. As the price of G_3 is currently p_3, bidder 0 must bid at least p_3 if it wants to win the slot. The number above a slot represents the index of the bidder winning the slot. Here, G_0 is currently won by bidder 1; bidder 0 is not winning any slot.

3.4 Bidding Strategies

As each bidder is self-interested, they tend to maximise their own surplus. In other words, each bidder will bid for the resource that maximises its value less the price it pays for those resources. The maximum surplus of a bidder *i* is defined as:

$$H_i = \max_{\mathbf{X}_i \subseteq \mathbf{G}} \left[v_i(\mathbf{X}_i) - \sum_{j \in \{r \mid G_r \in \mathbf{X}_i\}} p'_j \right] \qquad \text{Q 2}$$

Two bidding strategies are investigated in this project. They are the Straight Forward Bidding (SFB) strategy and the Straight Forward Bidding with Sunk Cost Awareness (SFBSCA) strategy. If a bidder uses the SFB strategy, p'_j equals the bid price of bidder *i* if it holds resource G_j; and equals the current price p_j (the price that it is going to bid) of the resource that it is not holding. When a bidder uses the SFBSCA strategy, p'_j equals the current price p_j for the resource that it is not holding and zero for the resources that it is already holding.

3.4.1 Straight Forward Bidding (SFB) Strategy

When a bidder uses the SFB strategy, it will bid for some slots if and only if:

1. It has not won enough resources to complete its job.
2. The total price of all resources that it bids (resources that it is currently winning and resources that it is going to bid) does not exceed its value.

To maximise it surplus, a bidder will always bid for the cheapest resources and will only bid at the current prices of the resources, which are the prices just enough for it to get those resources.

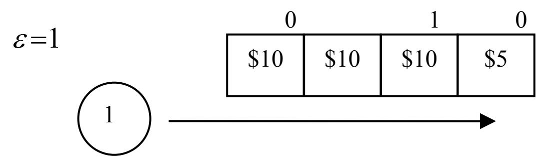


Figure 7. An example of a bidder using the SFB strategy.

An example of an auction with bidders using the SFB strategy is depicted in Figure 7. In this example bidder 1 has a value of $16 and it needs two resources to complete its job. There are four resources for bidding and the prices are $10, $10, $10 and $5. As seen from the diagram, the first and the forth resources are currently won by bidder 0, the third resource by bidder 1. The second resource G_1 is unallocated. Since bidder 1 needs two resources but it is only winning one and it has enough value to bid for G_3, it bids for that at a bid price of $5.

3.4.2 Straight Forward Bidding with Sunk Cost Awareness Bidding (SFBSCA) Strategy

When a bidder uses the SFBSCA strategy, it treats the bid price for the resources that it is winning as sunk cost. When it decides whether to bid for another resource, it only compares the price for the new resources with its value, disregarding the price of the resources that it is winning. At the first sight, this may sound counter-intuitive. However, as mentioned on page 29, the value of a bidder is zero if it cannot complete its job (if it does not get enough slots). In order to minimise its loss, the bidder may bid for a slot even if the sum of its current and previous bids exceeds its value (assuming that it can complete its job).

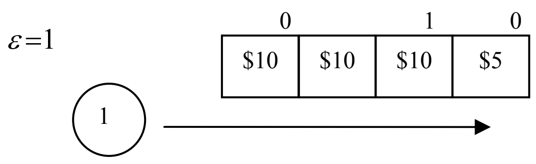


Figure 8. An example of a bidder using the SFBSCA strategy.

An example is illustrated in Figure 8. Bidder 1 has \$10 and it needs two resources. If bidder 1 uses the SFB strategy, it will stop bidding as it will need \$14 to bid for G_3 at \$5 as it is already having G_2 at \$9 (the bid price of bidder 1 for G_2 must have been \$9, thus making the current price to be \$10). However, if bidder 1 uses the SFBSCA strategy, it will disregard the amount that it previously placed for G_2 and bid for G_3 as long as its value is higher than or equal to the price of G_3. The rationale behind this is that if bidder 1 stops bidding at this point, it will not complete its job. It will receive zero return and it will lose \$9 (the price that it pays for G_2). However, if it continues to bid for G_3, it will get a return of \$10 for being able to complete its job and it pays \$14 (\$9 + \$5) for both resources. The net loss is only \$14 − \$10 = \$4, which is lower than the case had it used the SFB strategy.

It is show in simulations in Section 5.2 on page 39 and Section 5.3 on page 41 that the solutions produced by the MBSA with bidders using the SFBSCA strategy are better than that using the SFB strategy.

CHAPTER 4 PROBLEM DOMAIN

In this research, the resources comprise of machines and tools whereas the jobs comprise of products to be processed by the resources. The scheduling problem is thus to find a sequence in which the machines process the products. This is normally termed job shop scheduling. In this chapter, we first describe a job shop scheduling problem. We define a way to measure how good a solution to the scheduling problem is. We then present how the MBSA can be used in job shop scheduling. To benchmark the performance of the MBSA, we describe another algorithm, which may not be efficient, that can produce an optimum schedule.

4.1 System Description

In this research, a manufacturing environment that consists of a manufacturing factory and its customers is considered. The customers place orders and the manufacturing factory provides resources to process the orders. An order constitutes a job in scheduling terminology. In a real life environment, customers appear to submit orders at random intervals. That is, an order may arrive after an initial schedule is made and a new schedule is required. Each order has a *value*, which is the profit the customer gets from the order, and a deadline. The value of the order is zero if it is not processed before the deadline.

Most researches in job shop scheduling focused on variable length jobs and aimed to shorten the job earliness, tardiness or the makespan of the products. However, little attention has been given to job priority. In this research, fixed length jobs with varying priority are considered. As the jobs are of fixed length, the resources can then be regarded as discrete processing time slots of the machines in a factory. However, this can be generalised to a variable length job problem, in which a job requires one or more time slots. For the rest of this thesis, the term “slot” is used to represent a resource unit in the system.

From the manufacturer’s point of view, costs are incurred in providing these resources. The costs might come from machine depreciation cost, machine wear and tear, labour cost, electricity cost, and machine reconfiguration cost. Due to changes to these costs over time, a manufacturer may set different reserved prices for different slots. A slot will be allocated to an order only if the order is willing to pay a price higher than or equal to its reserved price.

4.2 System Performance Evaluation

A *solution* to a scheduling problem is defined as an assignment of the slots to the jobs. Intuitively, a good solution shall comprise of the following:

1. In the situation where there are more orders than available slots, not all orders can be processed. A solution that processes orders with higher values is better.
2. In the situation where there are more slots than orders, not all slots are assigned. A solution that assigns the orders to slots with lower reserved prices is better.

In view of these criteria, the global value function defined in Q 1 on page 29 is a suitable candidate to be used as the performance measure. To repeat, the global value is the sum of all the reserved priced of unallocated slots and all values of orders that can be completed.

4.3 Scheduling Approaches

In this research, two scheduling algorithms are used and compared. The first algorithm is the MBSA described in Chapter 3 on page 27. The second algorithm is a complete search, whose results are used to benchmark the performance of the MBSA.

4.3.1 The Market Based Scheduling Algorithm used in Manufacturing

To implement the MBSA in an HMS, the following mappings are used:

1. The machines are the auctioneers in auctions.
2. The orders are the bidders in auctions.
3. The time slots for machine operations are the resources which the bidders bid for.

Each slot has a reserved price. The orders submit bids to the machines for the slots. A machine accepts a bid if the bid price is equal to or higher than the current price of the slot. When all orders stop bidding, the auction close; the scheduling solution is assigning each slot to the order with the highest bid. In a dynamic system where orders arrive randomly, the closing criteria will be modified so that orders that arrive late will be able to bid for the slots. This is discussed in Section 7.1 on page 51.

4.3.2 Complete Search

To assess the performance of the MBSA, its solutions are compared to the best achievable solutions. A complete search evaluates all possible allocations of slots to orders and selects a solution that has the highest global value. A solution that attains the highest global value is termed as an *optimum solution* and the corresponding global value is called the *global optimum value*. It is noted that there might be more than one optimum solution but there is only one global optimum value for each scheduling problem.

CHAPTER 5 PERFORMANCE OF THE MARKET BASED ALGORITHM

In this chapter, the performance of the MBSA under different scheduling conditions is investigated. More specifically, the number of bidders, the number of resources, and the bidding strategy are varied.

5.1 Simulations Settings and Procedures

In each simulation, a set of reserved prices and bidders' values are first determined. Each bidder evaluates the slots to bid for in order to maximise its surplus and submits its bids to the auctioneer. The auctioneer processes the bids in random order. A bid for a slot will be accepted if it is equal to or higher than the current price. Here the random selection simulates the fact that bids arrive at different time in a real time physical system. After the bids are evaluated, the auctioneer notifies the current winner for each slot and broadcasts the current price of every slot. It is noted that the current prices of the slots can be either broadcasted or enquired by the bidders. A bidder bids again if it has not got enough slots and it has enough value. When all bidders stop bidding, the global value of the solution is calculated.

The conditions and the parameters used in the simulations in this chapter are listed below.

1. Each bidder requires a number of slots.
2. If a bidder is able to get enough slots, it will get some value. If the bidder gets less than the required number of slots, it gets zero value.

3. Without loss of generality, the reserved prices, the bid prices and the bidders' values assume integer values.
4. In each simulation run, the reserved price of each slot is randomly generated from $\{1, 2, ..., 10\}$.
5. In each simulation, a bidder's value is randomly generated from $\{1, 2, ..., 10k\}$, where k is the number of slot that the bidder needs.

With the above conditions, the following scenarios are simulated and the results are shown in Section 5.2 on page 39 and Section 5.3 on page 41.

1. There are two bidders, $M = 2$. Each bidder requires two slots and each uses the SFB strategy. The number of available slots changes from $N = 2$ to $N = 10$. For each value of N, 10000 auctions are simulated. The reserved prices and the bidders' values are different and randomly generated for each auction.
2. (1) is repeated with bidders using the SFBSCA strategy.
3. The number of bidders changes from $M = 2$ to $M = 6$. Each bidder requires two slots and each bidder uses the SFB strategy. The number of available slots is kept constant, which is six slots. For each value of M, 10000 auctions are simulated. The reserved prices and the bidders' values are randomly generated for each auction.
4. (3) is repeated with bidders using the SFBSCA strategy.

5.2 Performance Comparison with Varying Total Available Slots

The results of simulations (1) and (2) are shown in Figure 9 and Figure 10. Figure 9 shows the performance of the MBSA in terms of the global value. The values are averaged over 10000 simulation samples. The difference between the global values of the solutions produced by the MBSA and the global optimum values is plotted in Figure 10. The difference is normalised to the global optimum value.

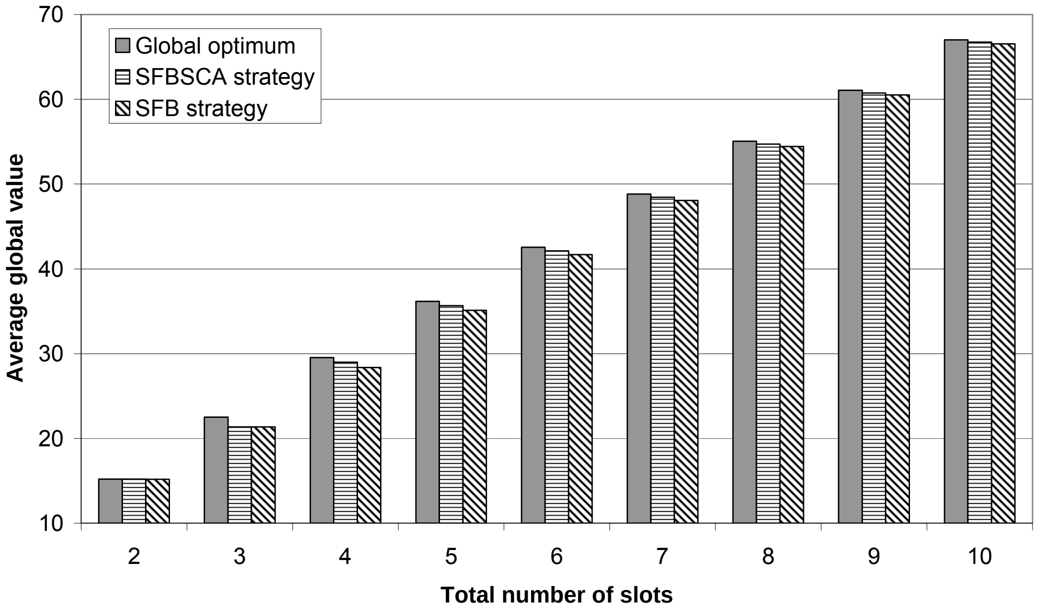


Figure 9. A graph showing the performance of the MBSA with varying number of slots.

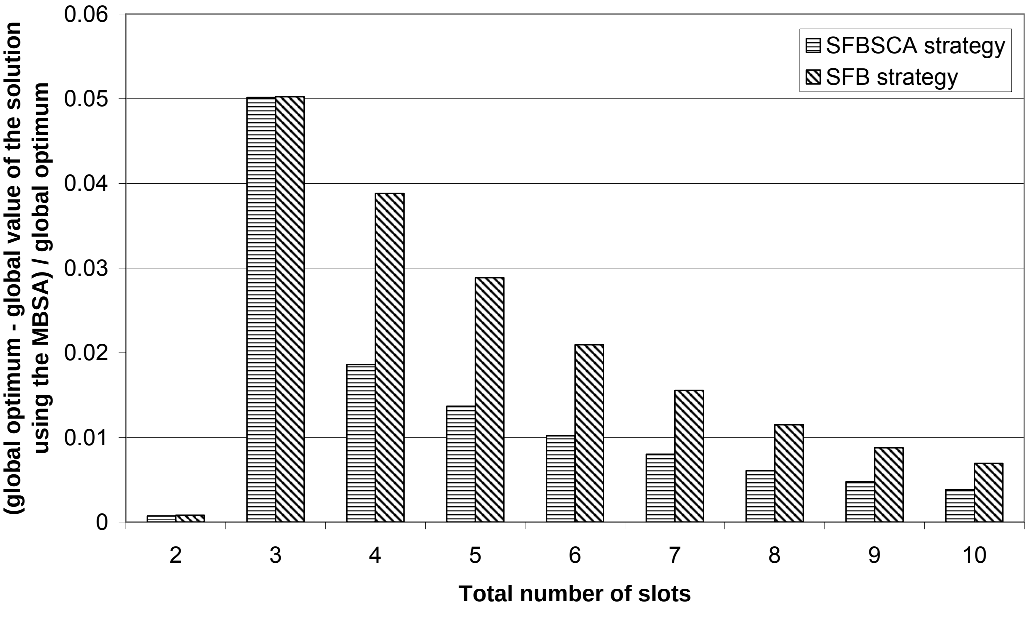


Figure 10. A graph showing the difference in the global optimum value and the global value of a solution produced by the MBSA, normalised to the global optimum value.

5.3 Performance Comparison with Varying Number of Agents

The results of simulations (3) and (4) are shown in Figure 11 and Figure 12.

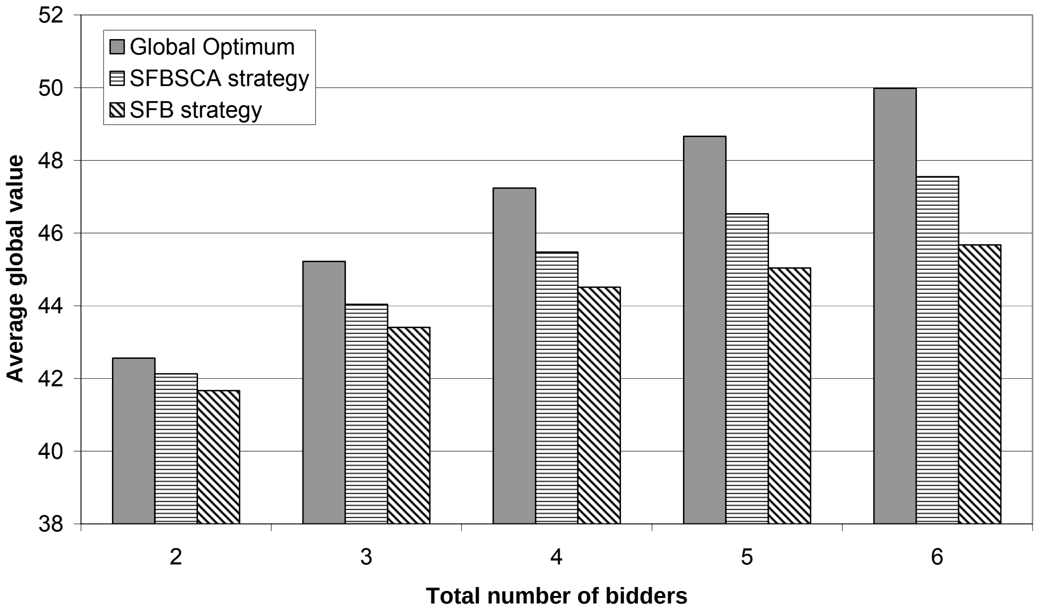


Figure 11. A graph showing the performance of the MBSA with varying number of bidders.

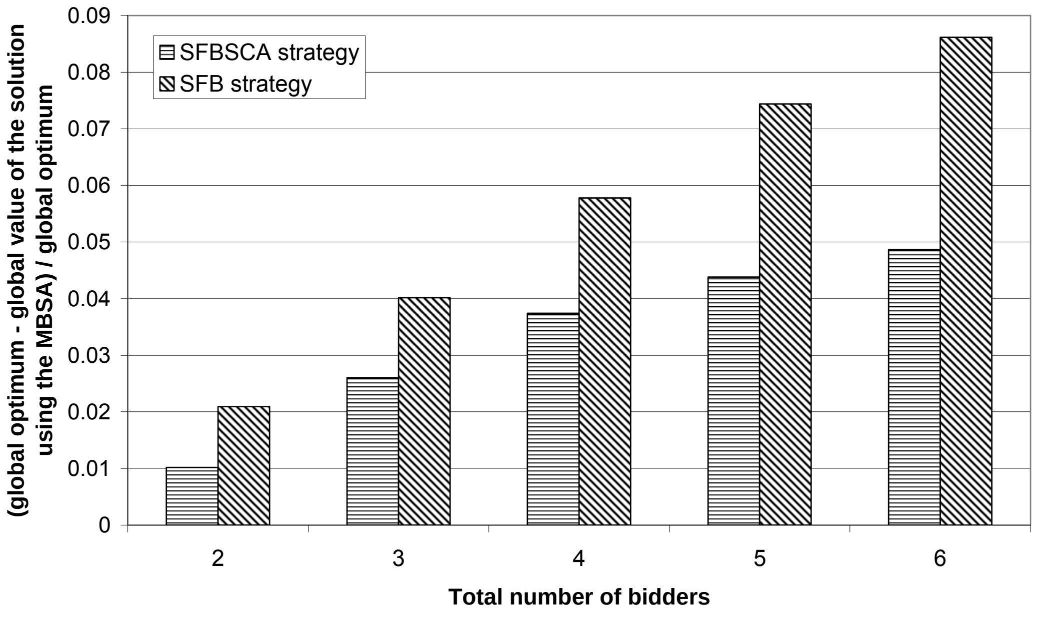


Figure 12. A graph showing the difference in the global optimum value and the global value of a solution achieved by the MBSA, normalised to the global optimum value.

5.4 Discussions on the Performance of the Market Based Scheduling Algorithm

It is observed that the performance of the MBSA is better when the bidders use the SFBSCA strategy compared to the case where the bidders use the SFB strategy. This is seen from the graphs that the average global value of the solution when the SFBSCA strategy is used is higher than that when the SFB strategy is used.

From Figure 9, it is noted that when there are only two slots and two bidders, the solution when the bidders use the SFBSCA strategy is the same as that using the SFB strategy. This is because when there are two bidders with each requiring two slots, a bidder either bids the two available slots or quits. Hence, at any time, the two slots are either both bid by a bidder or both not being bid for. There is therefore no situation where a bidder needs to consider the sunk cost.

It can be seen from Figure 10 that the performance of the MBSA improves when there are more resources in the system. When there are more slots, there are fewer instances where the bidders fight for a slot. When a bidder requires two slots and ends up getting only one, the global value is affected. This is because the reserved price of that slot cannot be summed to the global value and the value of the bidder (that gets only one slot) also cannot be summed to the global value. A better assignment would be to leave that slot unallocated.

Keeping the number of total slots constant, we see that the performance of the MBSA degrades when the number of bidders in the system increases. If there are more bidders in the system, the chance of the slots being allocated to different bidders (and some of them do not get enough slots) is higher. Hence, the global value of the solution is lower.

Hence, the performance of the MBSA performance is better when the ratio of slots to bidders increases. A manufacturing company should be aware that when the resource to job ratio decreases, the performance of the MBSA may start to degrade. One way to avoid this is to design the reserved price of the slots such that optimum results will always be produced by the MBSA. An analysis of the effect of the reserved price on the optimality of the solution is presented in the next chapter.

CHAPTER 6 WHEN THE MARKET BASED ALGORITHM UNDER-PERFORMS

When one uses the MBSA, one may be interested in the worst possible performance of the algorithm. Also, one may wish to adjust parameters of the auctions, such as the reserved prices of the resources, in order to achieve a better result. In this chapter, the lower bound of the global value of solutions produced by the MBSA is derived. After that, conditions under which the MBSA performs badly are investigated.

6.1 Upper Bound of Suboptimality

6.1.1 Bidders with Single Slot Jobs

In the scheduling problem where each bidder requires only one slot, Wellman showed that the global value of a solution produced by the MBSA is at most $\kappa\varepsilon(1+\kappa)$ lower than the global optimum value [6]. Here, $\kappa = \min[N, M]$ where N is the total number of slots and M is the total number of bidders.

6.1.2 Bidders with Two-Slot Jobs

In this section, we derive the lower bound of the global value of a solution produced by the MBSA when there are two bidders and each bidder required two slots to complete their jobs. The jobs are non-preemptive. That means that the bidders do not require slots in sequence.

We define the vector $\mathbf{f} = [f_0 \ f_1 \ \dots \ f_{N-1}]$ to be the allocation of slots to the bidders where slot 0 is assigned to bidder f_0 and so on. If $f_i = -1$, slot i is unallocated. The following system is considered:

1. There are two bidders, bidder 0 and bidder 1, with values v_0 and v_1 respectively. In addition, $v_0 \leq v_1$. Bidder 1 has a value higher or same as bidder 0.
2. Each bidder requires two slots to complete its job.
3. Each bidder uses the SFBSCA strategy.
4. There are four available time slots: slot 0, slot 1, slot 2 and slot 3, each with the reserved price of $q_0, q_1, q_2, \text{and}\, q_3$ respectively. Without loss of generality, the time slots are in the order of non-increasing reserved price, that is $q_0 \geq q_1 \geq q_2 \geq q_3$. The lowest reserved value is $q_{\min} = q_3$ and the highest reserved value is $q_{\max} = q_0$.

To find the global value of a solution, we first break up the problem into different categories. The categories are divided according to the optimum solution of the problem. This way of dividing the problem makes the derivation of the global value systematic and relatively easy. After dividing the problem domain into different categories, we look into each category and find out how *suboptimality* may arise. This is done by comparing the optimum solution with any suboptimum solutions that may be generated using the MBSA. Suboptimality of a solution is defined as:

Suboptimality = Global optimum value − Global value of a solution

There are three categories of optimum solutions:

1. The optimum solution is $\mathbf{f} = [-1 \; -1 \; -1 \; -1]$ if the highest global value is achieved by not allocating any slot to any bidder.
2. The optimum solution is $\mathbf{f} = [1 \; 1 \; 0 \; 0]$ or any permutation if the highest global value achieved by allocating all four slots to both bidders.
3. The optimum solution is $\mathbf{f} = [-1 \; -1 \; 1 \; 1]$ if the highest global value is achieved by allocating the two cheapest slots to bidder 1.

A solution is a suboptimum solution if its global value is strictly less than the global optimum value. We give an example of how a suboptimum solution may arise in Appendix A. In Appendix B, we analyse how a suboptimum solution may result using the MBSA. In addition, we derive the upper bound of the suboptimality in each category.

The results are summarised in Table 2. If a scheduling problem falls into category (a), (b), or (g), the MBSA will always produce an optimum solution. A suboptimum solution may occur if the problem falls into category (h). In that case, assuming the price increment for each round of bidding is small, that is $\varepsilon < \frac{q_{\max}}{2}$ and $\varepsilon < q_{\min}$, the upper bound of the suboptimality of a solution using the MBSA is $q_{\max}$.

Table 2. A summary of the upper bounds of the suboptimality of solutions produced by the MBSA.

	Optimum solution (or any permutations)	Suboptimum solution (or any permutations)	Maximum suboptimality
(a)	[−1 −1 −1 −1]	No suboptimum solution	−
(b)	[1 1 0 0]	No suboptimum solution	−
(c)	[−1 0 1 1] is not an optimum solution	−	−
(d)	[−1 1 0 0] is not an optimum solution	−	−
(e)	[−1 −1 −1 0] is not an optimum solution	−	−
(f)	[−1 −1 −1 1] is not an optimum solution	−	−
(g)	[−1 −1 0 0] - when $v_1 > v_0$, it is not a global solution - when $v_1 = v_0$, refer to (h)	−	−
(h)	[−1 −1 1 1]	[−1 0 1 1]	$q_{\max}$
		[1 1 0 0]	$q_{\max} - q_{\min} + \varepsilon$
		[−1 −1 0 0]	$v_1 - v_0$ $(< 2\varepsilon)$
		[−1 1 0 0] is not possible	−

6.2 The Effect of Bidders' Values on the Optimality

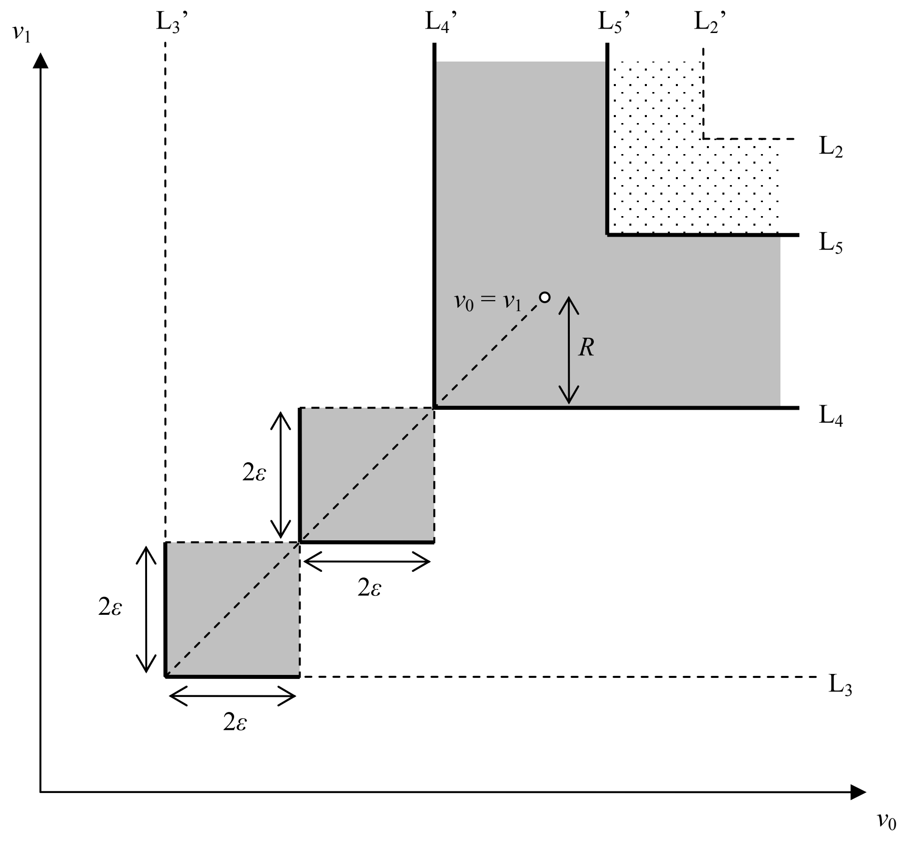
In the previous section, we derived the upper bound of the suboptimality of solutions produced by the MBSA. Besides the bound, one may be interested in knowing under what circumstances the MBSA always gives an optimum solution. In this section, we investigate the combinations of reserved prices and bidders' values that make the MBSA produces optimum solutions.

The following system is considered.

1. There are four or more slots. The four lowest reserved prices of the slots are $q_{\mathrm{A}}, q_{\mathrm{B}}, q_{\mathrm{C}}, \text{and}\, q_{\mathrm{D}}$ where $q_{\mathrm{A}} \le q_{\mathrm{B}} \le q_{\mathrm{C}} \le q_{\mathrm{D}}$.
2. There are two bidders, each requires two slots. The values of the bidders are v_0 and v_1.

We found that for a given set of reserved prices, the bidders' values space can be divided into a few regions which each region has one of the following characteristics:

1. The MBSA always produces an optimum solution.
2. The MBSA always produces a suboptimum solution.
3. The MBSA sometimes produces an optimum solution depending on the sequence in which the order bids and the slots that a bidder bids when there are more than two slots of same price.



▨ Suboptimality may sometimes occur (include L_4; exclude L_5 and dotted lines)

⁙ Suboptimality will always occur (include L_5; exclude L_2)

$$L_2 : v_1 = q_C + q_D$$

$$L_3 : v_1 = q_A + q_B$$

$$L_4 : v_1 = q_A + q_B + 2(k-1)\varepsilon$$

$$L_5 : v_1 = q_A + q_C + l\varepsilon$$

$$k = \left\lceil \frac{q_C - q_B}{\varepsilon} \right\rceil$$

$$l = \left\lfloor \frac{q_C - q_B}{\varepsilon} \right\rfloor + 1$$

$$R = q_C - q_B + (2-k)\varepsilon$$

Figure 13. A graph showing regions in which suboptimality will always/will never/may sometimes occur.

The regions are shown in Figure 13. If the bidders' values fall into the grey regions, the MBSA may produce suboptimum solutions. If the bidders' values fall into the dotted region, suboptimum solutions will always occur. Otherwise, the algorithm will always produce optimum solutions. The full derivation of the graph can be found in Appendix C.

In practice, a manufacturing factory may wish to reduce the grey region and the dotted region by adjusting the lines $\mathrm{L}_2, \mathrm{L}_3, \mathrm{L}_4, \text{and } \mathrm{L}_5$. This can be done by changing the four lowest reserved prices of the slots. Doing so, it is possible to reduce the possibility of bidders' values (which the manufacturing company has no control) falling into the grey region or the dotted region; and thus reduces the possibility of suboptimum solutions. To guarantee that the MBSA always produces optimum solutions, we can set $q_\mathrm{A} + q_\mathrm{B} = q_\mathrm{C} + q_\mathrm{D}$ or $q_\mathrm{A} = q_\mathrm{B} = q_\mathrm{C} = q_\mathrm{D}$. Otherwise, the MBSA may produce suboptimum solutions.

CHAPTER 7 IMPLEMENTATION OF THE MARKET BASED ALGORITHM

In this chapter, we present the issues and findings of implementing the MBSA in Robot Cell Test Harness and in Cambridge Auto-ID Lab which implements an HMS. Robot Cell Test Harness is a simulated environment designed according to Cambridge Auto-ID Lab. We propose modifications to the original MBSA to suit a manufacturing environment. This is followed by a brief description of the Auto-ID Lab set up and the simulation. We present the implementation results on both simulations and on a physical HMS system. At the end of this chapter, we present our investigations of the effect of varying slot period on the performance of the MBSA.

7.1 Adapting the Market Based Scheduling Algorithm in an HMS

In this section, we present our proposed modifications to the original MBSA for it to be implemented in a real time manufacturing system.

7.1.1 Continuous Auction

For a manufacturing system in general, orders from customers arrive in a random order, even after an initial schedule is made. This requires the auction for a slot to be "open" until some time before a job is processed in that slot. In this case, a schedule can constantly be updated with new information, which is the winning bidder for each slot. The question now is how early an auction for a slot must be closed prior to that slot.

Considering a typical manufacturing environment where products are stored at a distance from the machines, a product must travel to the vicinity of the machine at certain time before its scheduled slot for operation. The travelling time may vary depending on its location in the storage area and the position of the machine with respect to the storage area. Supposing the maximum time it takes to transport a product from the storage area to the vicinity of the machine is $T_{travelling}$, a product may not be able to get to the machine if it bids for a slot within $T_{travelling}$ before the time of operation.

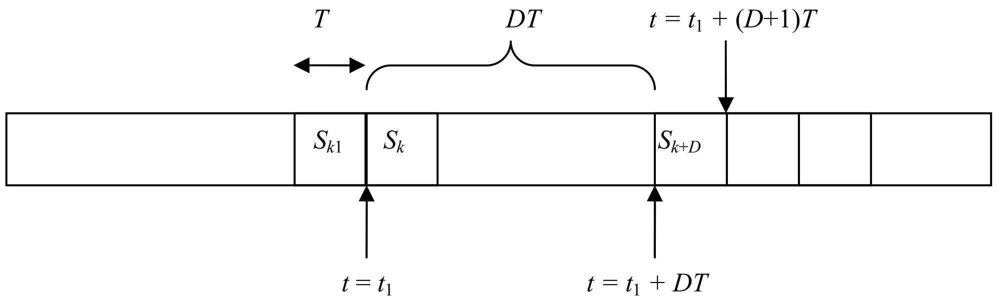


Figure 14. A job can only bid for slots *DT* time ahead.

In order to make sure that a product arrives at the vicinity of the machine before its slot, it must bid for a slot that is at least *D* slots ahead. Referring to Figure 14, at time between $t_1 - T \le t < t_1$, an order can only bid for slots S_{k+D} onwards. This also means that the auction for slot S_{k+D} closes at t_1, which is DT before the starting of the slot (at $t = t_1 + DT$). *D* must satisfy the following condition:

$$DT \ge T_{travelling} \qquad \text{Q 3}$$

Here, T is the period of a slot.

If every product has a different travelling time from the storage area to the vicinity of the machine, a product may not be queued at the machine according to the schedule. In general, the product may be placed in a temporary storage place near the machine.

Referring to Figure 14, at $t_1 \le t < t_1 + T$, the product scheduled in slots S_k to S_{k+D} (inclusive) may have arrived and the product processed in S_{k-1} may not have left the vicinity of the machine. We assume that any product processed on slots earlier than S_{k-1} have left. Hence the maximum number of product cluttered at the vicinity of a machine at any one time, Q , given by the following equation:

$$Q = D + 2 \qquad \text{Q 4}$$

This poses a physical constraint on D .

7.1.2 Selection of the Time Slot Period

In this research, a constant slot period is considered. A constant period is useful when an order bids for a slot and it needs to know when the slot is to be executed so as not to miss its deadline. The choice of the slot period is important. Choosing a slot period that is too small might cause an operation to exceed its slot. This might affect the operations in subsequent slots. Choosing a slots period that is too large causes a waste of resources as more slots could have been allocated and more jobs could have been processed.

The slot period must be at least equal to the operation time of a product. The operation time is different for different manufacturing processes. In general, the operation time depends on the speed of the machine and the number of products at the machine vicinity, Q . This can be expressed by the following inequality:

$$T \ge T_{operation}(Q) \qquad \text{Q 5}$$

Here, $T_{operation}(Q)$ is the operation time and it may be dependent on Q .

As we would like to pack as many items as possible (efficient use of resources) in a given period of time, the design problem is to minimise T with Q 3 and Q 5 as constraints. The fixed variables in these two equations is $T_{travelling}$. It can be measured directly from the system. One method that can be used to minimise T is:

1. Measure $T_{travelling}$.
2. For $D = 1, 2, 3, \ldots$, up to the limit constrained by Q, measure $T_{operation}(Q)$.
3. For each D, calculate the minimum value of T that satisfies Q 3 and Q 5.
4. Choose D of which the minimum of T in (3) is the smallest.

7.2 Cambridge Auto-ID Lab

Cambridge Auto-ID Lab is a testbed for an HMS. The system is implemented using an MAS. Each order (an order is a customer's request for a box to be packed with certain items in certain orientation) and each robot (which constitute the resource) has a software agent that controls it and interacts with other agents.

Figure 15 depicts the robot pick-and-place setup in Cambridge Auto-ID Lab. The gift boxes are placed on shuttles that travel on the track loops. The shuttles constantly travel round the main loop. When an order arrives, a particular gift box is assigned to the order. The aim of the system is to pack gift boxes before their deadlines. One packing operation involves the following steps:

1. The shuttle that carries the box moves into a side loop.
2. It stops at the docking station.
3. The robot picks items from the storage area and places them into the gift box.
4. After a box is packed, the shuttle travels back into the main loop.

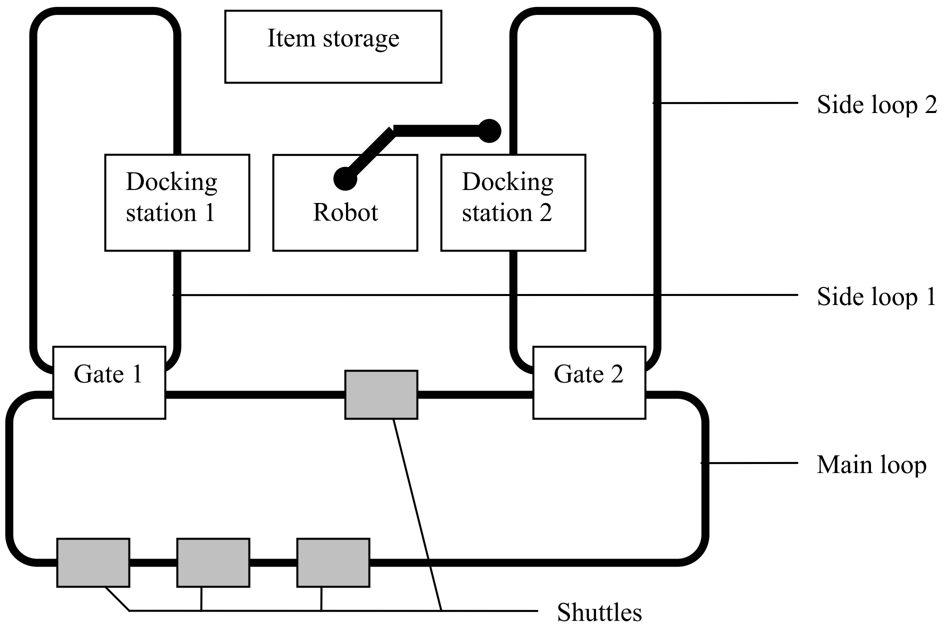


Figure 15. Robot pick-and-place setup in Cambridge Auto-ID Lab.

In such a set up, the main loop represent the storage area in which a box is "stored" and the side loop represents the vicinity of packing operation. $T_{travelling}$, which is the maximum time for a shuttle to travel from anywhere in the main loop into a side loop, is the time for a shuttle travels once round the main loop, T_{main_loop}. Before the execution of an order, the shuttle must be left in the side loop and cannot be positioned in the docking station. This is because there might be another order that wants to use the docking station. Hence the operation time, $T_{operation}(Q)$, consists of the time taken for a shuttle to travel from the side loop to the docking station plus the time taken for the robot to pack the box. Taking the worse case situation, the time taken for a shuttle to travel from the side loop to the docking station is the time for it to travel once round the side loop. Hence,

$$T_{operation}(Q) = T_{side\_loop}(Q) + T_{packing} \qquad \text{Q 6}$$

Here, $T_{side\_loop}(Q)$ is the time for a shuttle to travel once round the side loop when there are Q shuttles in the side loop and $T_{packing}$ is the time taken for the robot to pack a box. $T_{packing}$ is independent of Q.

7.3 Implementation on Robot Test Cell Harness

7.3.1 Simulation Setup

The implementation of the MBSA is first tested on Robot Test Cell Harness, a simulated HMS designed according to the setup in Figure 15. The interface between the scheduling program and the simulation is described in Appendix D.

Table 3. Reserved price of slots used in simulation.

Slot index	0	1	2	3	4	5	6
Reserved price	9	5	2	2	3	4	5

Table 4. Configurations of orders used in simulation.

Order number	Value	Arrival time	Deadline
0	4	1	10
1	5	0	10
2	4	1	10
3	7	2	3

A scheduling problem of a system with seven slots and four orders are simulated. The reserved prices of the slots are shown in Table 3 and the value, arrival time, and deadline of the orders are shown in Table 4. The values of the reserved prices are chosen in such a way that there are two cheap slots so that orders that arrive late will bid for them. This allows us to show certain behaviours of the algorithm. The orders' values are chosen in such a way that they have enough value to bid for the slots. There is a rush order (order 3) which arrives late and has an early deadline. In this simulation, for simplicity, we assume that an order arrives just before the beginning of a slot and the deadline is at the end of a slot. For example order 0 arrives just before the beginning of slot 1 and its deadline is at the end of slot 10. Also, time taken for the bidding to reach equilibrium is assumed to be negligible.

7.3.2 Implementation Results on Robot Test Cell Harness

7.3.2.1 Calculation of Slot Period T and D

Based on 500 simulation runs, the time for a shuttle to go once round the main loop, time for a shuttle to go once round the side loop and the pacing time are recorded and the maximum value out of the 500 simulation is as shown below. The time scale is based on unit simulation time.

$$T_{main\_loop} = 53s$$

$$T_{packing} = 22s$$

Table 5. Time taken for a shuttle to travel once round the side loop in simulation.

Q	$T_{side\_loop}(Q)$
1	19s
2	23s
3	33s
4	38s
5	49s

From the data above, we find the minimum T based on the following constraints:

$$T \geq \frac{T_{main\_loop}}{D} \qquad \text{Q 7}$$

$$T \geq T_{side\_loop}(Q) + T_{packing} \qquad \text{Q 8}$$

and Q 4 on page 53.

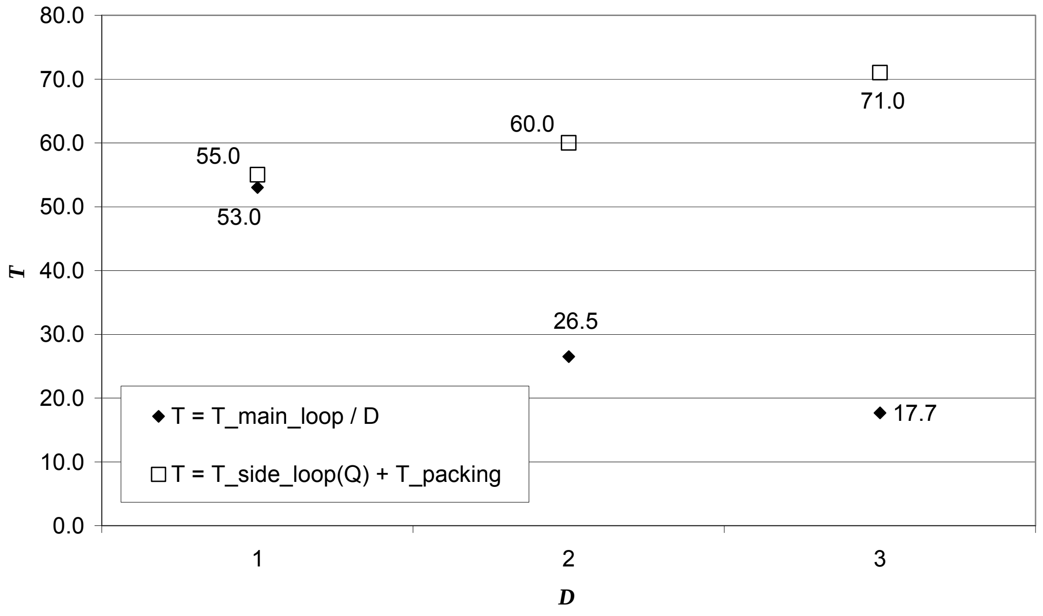


Figure 16. The constraints of slot period for simulation.

Figure 16 shows the constraints of Q 7 and Q 8 plotted on a graph. We see that the minimum T is 55s at $D = 1$.

7.3.2.2 Simulation Outputs

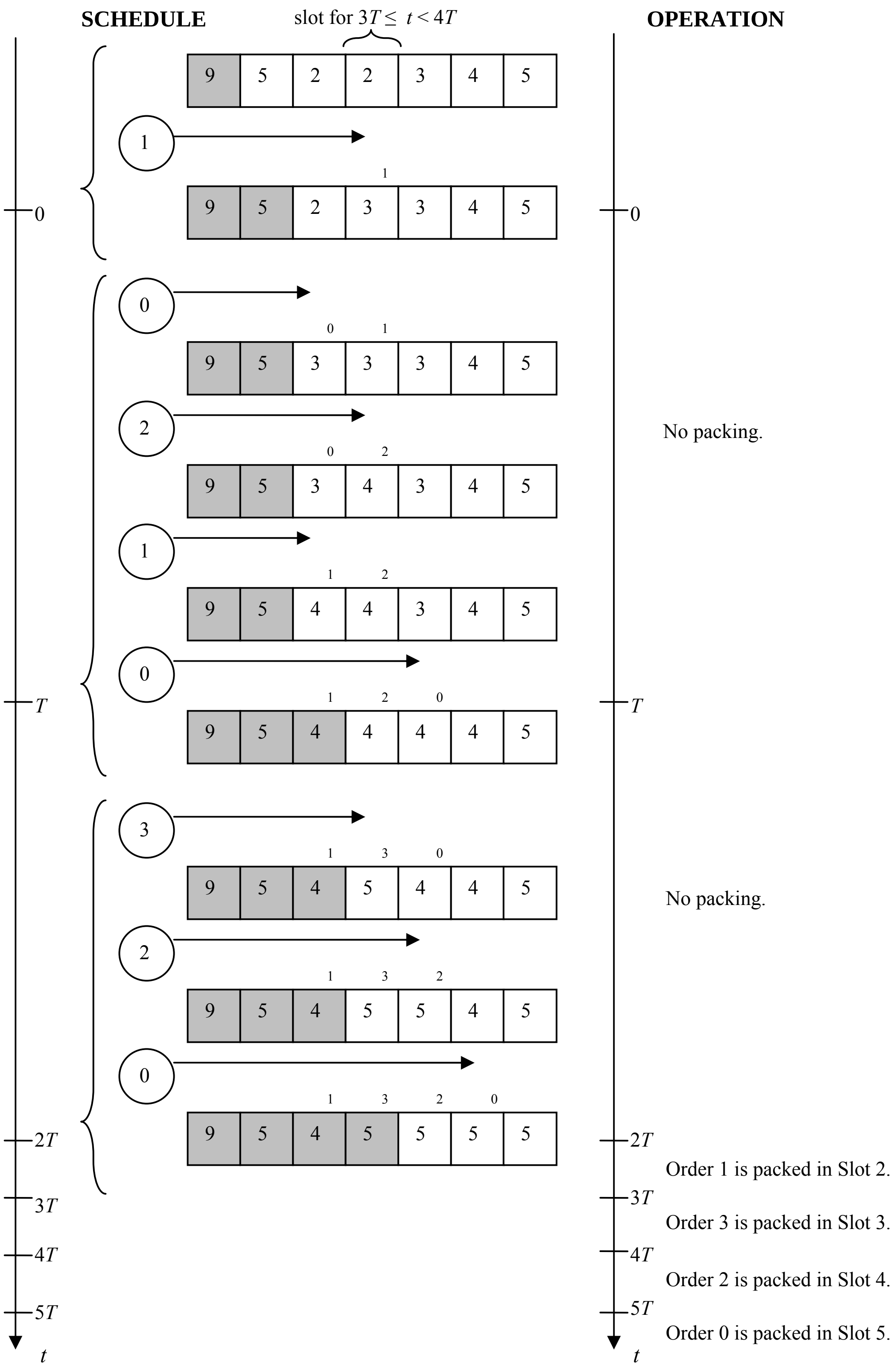


Figure 17. Extracts of auctions and packing operations in simulations.

Figure 17 shows a simulation run of an auction and packing operations. The left column shows the bidding process and the right column shows the packing operations. The vertical axis indicates the evolution of time (not drawn to scale). As stated earlier, all bidding reaches equilibrium at just before the beginning of a slot and the time taken is negligible compared to one slot period. Each packing operation is executed and completed within each time slot. Slots in grey are slots that are not available for bidding, as the auction for those slots are closed.

Just before the beginning of time slot 0, only order 1 has arrived. As $D = 1$, only slots 1 to 6 are available for bidding. Slots 2 and 3 (both with price \$2) are the cheapest, order 1 selects slot 3 by random. This increases the price of slot 3 to \$3. Just before $t = T$, orders 0 and 2 arrive and join the auction. As the cheapest slot now is slot 2, both order 0 and 2 bid for that slot. As indicated in the diagram, the bid from order 0 arrives first (the arrival of bids are simulated in random) and it gets slot 2. Order 2's bid, which arrives late, will be rejected (not shown in the diagram). So it bids for the next slot 3, which is initially owned by order 1. As order 1 now loses its slot, it bids for slot 2. Order 0, which originally owned slot 2, is being outbid and it bids for slot 4. At this point, the auction reaches equilibrium as no order places further bids. The bidding starts again when order 3 arrives before $t = 2T$.

7.4 Implementation on the Physical Cell in Cambridge Auto-ID Lab

As the simulated environment, Robot Test Cell Harness, is designed to be very similar to the actual hardware set up in Cambridge Auto-ID Lab, little problem is encountered when the scheduling program is migrated from the simulated environment to the actual lab environment. This implementation phase serves to verify the results obtained in the simulations.

7.4.1 Scenarios to be Tested

To verify the results of implementation on simulations, a similar scheduling problem with orders' information as stated in Table 3 on page 56 and reserved prices as stated in Table 4 on page 56 is tested. The MBSA is tested in the following two different lab configurations:

1. One docking station (docking station 1) and one robot.
2. Two docking stations (docking station 1 and docking station 2) and one robot.

7.4.2 Modifications on the Algorithm for Two Docking Stations

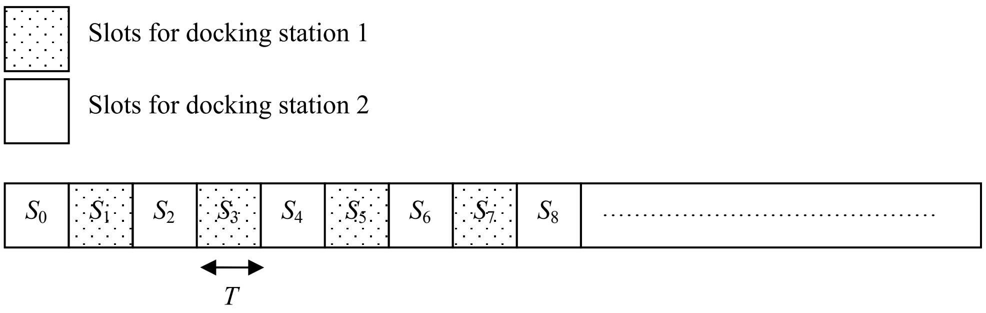


Figure 18. Assignment of slots for two docking stations.

In the case where two docking stations and one robot are in use, the scheduling algorithm will have to be modified. Since there is only one robot, the schedules for both docking stations are dependent. At any time, only shuttle in either docking station will be packed. One possible way of allocating tasks on the docking station is to have them operating at alternate slots. The odd slots are assigned to docking station 1 and the even slots are assigned to docking station 2. In this case, the maximum number of shuttles in a side loop at any time will be different from that in Q 4. Appendix F shows that Q for the case of two docking station is

$$Q = \begin{cases} \dfrac{D+3}{2} & \text{, if } D = 2n-1 \\ \dfrac{D+2}{2} & \text{, if } D = 2n \end{cases} \qquad n \in \mathbb{Z}^+ \qquad \text{Q 9}$$

7.4.3 Implementation Results on Physical Hardware

7.4.3.1 Calculation of T and D

As the values of T_{main_loop}, $T_{side\_loop}(Q)$ and $T_{packing}$ in the lab set up are different from that in simulations, the choice for T and D have to be re-evaluated. Also, the timing information for side loop 2 has to be considered. Below are the largest readings taken from ten samples.

$$T_{main\_loop} = 61.3\text{s}$$

$$T_{packing} = 69.6\text{s}$$

Table 6. Time taken for a shuttle to travel once round a side loop in the physical cell.

Q	$T_{side\_loop}(Q)$	
	Side loop 1	Side loop 2
1	29.5s	26.5s
2	32.5s	31.6s
3	34.1s	31.0s
4	34.6s	–

For the case of one docking station, the calculation for T and D follows that in Section 7.3.2.1 on page 57.

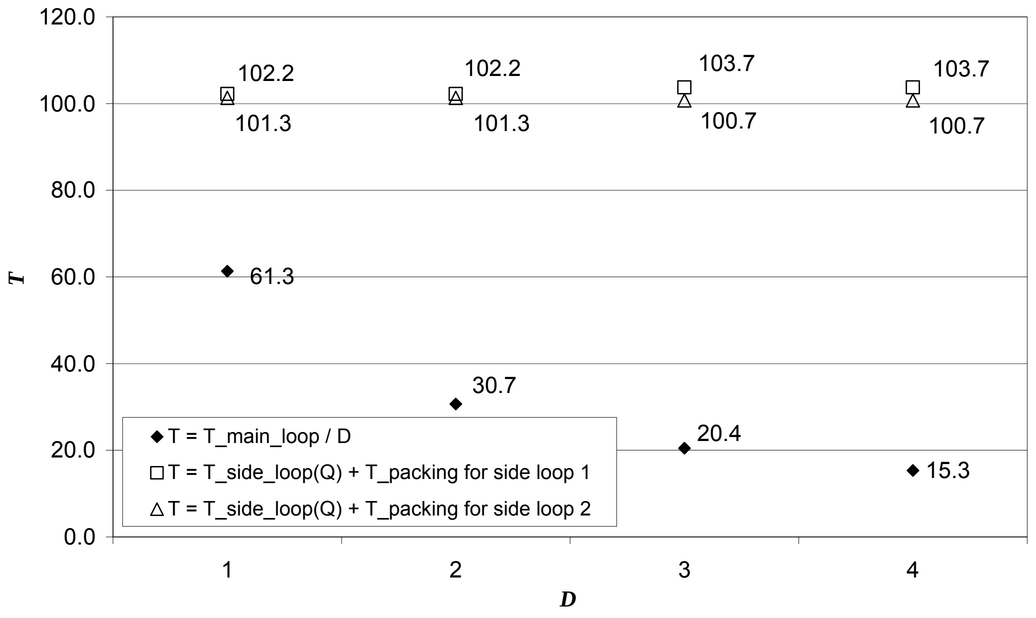


Figure 19. The constraints of slot period for lab setup with two docking stations.

For the case of two docking stations, the constraint Q 8 on page 58 for both the docking stations is taken in account. Also Q 4 on page 53 is replaced by Q 9 on page 62. Figure 19 shows the constraints of T plotted on a graph. We see that the minimum T is 102.2s at $D = 1 \text{or} 2$. $D = 1$ is preferred as the later the auctions close, the more flexible the system is. Table 7 show the choice of slot period, T, and D.

Table 7. The choice of *T* and *D* for the physical cell.

	One docking station	Two docking stations
T	103.7s	102.2s
D	1	1

7.4.3.2 The Schedule Produced by the Market Based Scheduling Algorithm

As the same schedule problem is tested on simulations and on physical hardware, the MBSA produces the same solution for both cases, which is depicted in Figure 17 on page 59. The solution is reproduced in Table 8 for easier reference.

Table 8. Slots assignment to the orders.

Slot index	Order to be processed
0	Unallocated
1	Unallocated
2	1
3	3
4	2
5	0
6	Unallocated

7.4.3.3 Implementation Results on Physical Hardware with One Docking Station

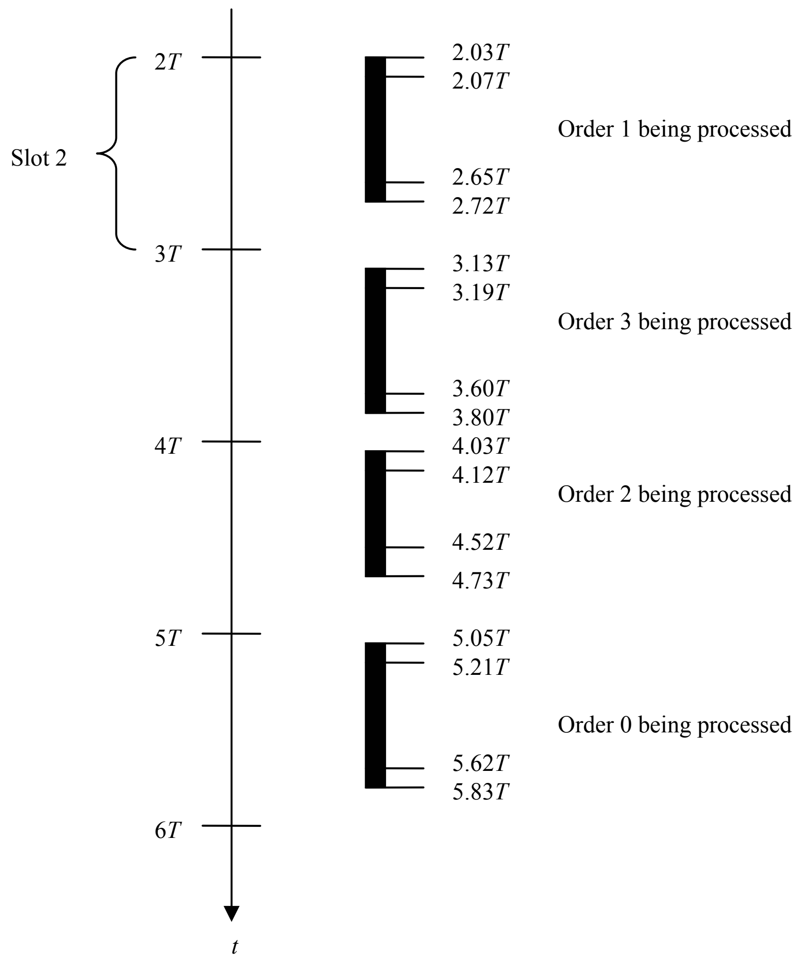


Figure 20. Time diagram showing the time which the robot starts/finishes packing the boxes in a system with one docking station.

Figure 20 depicts the actual operations that took place in three sample runs. As indicated in the diagram, time between $2T \leq t < 3T$ belongs to slot 2 and so forth. The thick vertical bars indicate the packing operations. Among the three runs, the earliest packing operation starts at time $t = 2.03T$ and the packing operation ends latest at time $t = 2.72T$. The packing start time and finish time for other orders are shown in a similar manner.

7.4.3.4 Implementation Results on Physical Hardware with Two Docking Stations

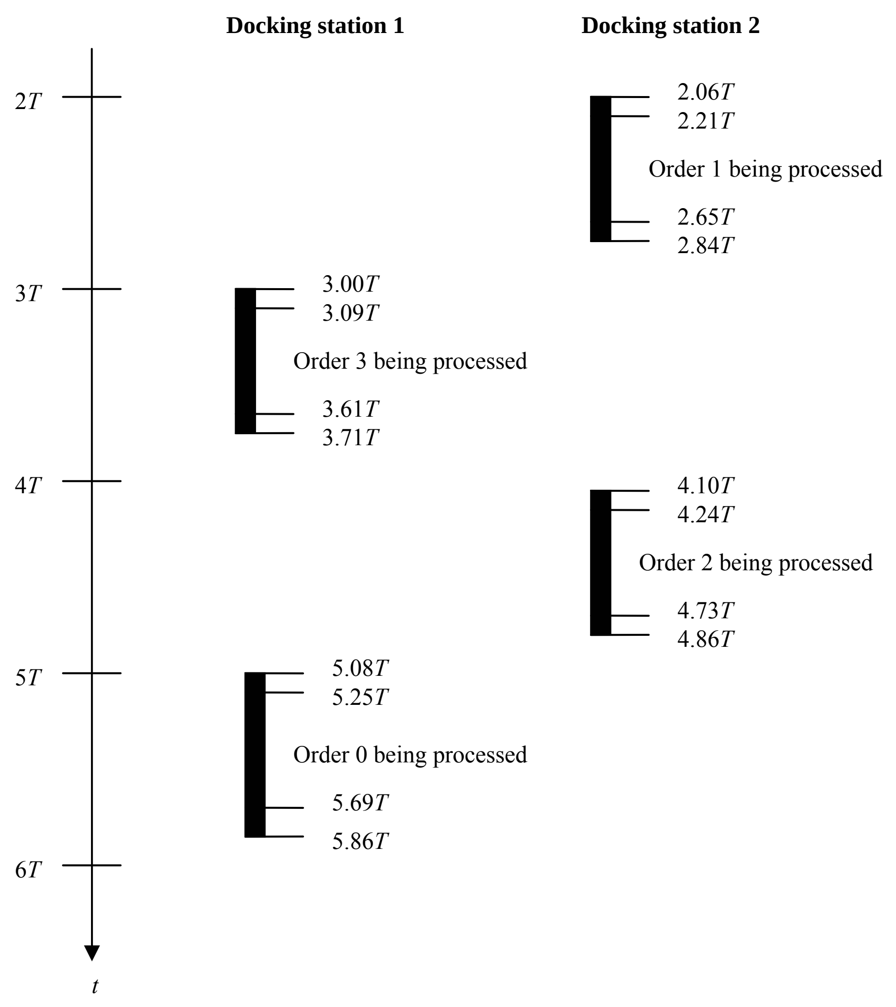


Figure 21. Time diagram showing the time which the robot starts/finishes packing the boxes in a system with two docking stations.

Figure 21 shows the timing for packing operations when two docking stations are in use. The thick lines in the middle column represent packing operations done in docking station 1 and the right most column docking station 2.

7.5 Discussions on the Implementation Results

7.5.1 The Ability of the Market Based Algorithm to Perform Scheduling in a Dynamic Environment

The MBSA is able to perform scheduling in a dynamic environment. This is done by using a continuous auction that we proposed. This means that the auction for a slot can still accept bids up to some time before its execution. It allows an order to arrive at any time, join the auction and bid for the slots.

Referring to Figure 17 on page 59, only order 1 has arrived just before $t = 0$. A schedule is made with order 1 being assigned to slot 3. Later, just before $t = T$, new orders (orders 0 and 2) arrive and a new schedule is formed.

7.5.2 Rescheduling Time and Quality

The rescheduling process is relatively simple using the MBSA. For the case of new order arrival, the process involves:

1. New order bids for its desired slots.
2. If the slot is already owned by an order, the order that is outbid bids for another slot.
3. The bidding stops when:
    a. All orders get their slots, or
    b. All orders that do not get their slots have insufficient value to bid.

Referring to Figure 17, just before $t = 2T$, order 3 arrives and the rescheduling is done in three bidding steps:

1. Order 3 bids for slot 3.

2. Order 2 bids for slot 4.
3. Order 0 bids for slot 5.

We compare the solution produced by the MBSA with that of the optimum solution. After $t = 2T$, all orders have arrived and no more rescheduling is done. The MBSA produces a schedule that allocates slots 2, 3, 4, and 5 to orders 1, 3, 2, and 0 respectively. The global value for this solution is \$39.

In determining the optimum solution, we assume that all orders' information is known at the beginning (this will not happen in a real system). If all information is available before $t = 0$, an optimum solution is to assign slots 2, 3, 4, and 5 to orders 1, 3, 2, and 0 respectively (taking into account the restriction of slots that an order can choose according to its arrival time). The global optimum value is \$39. We see that the MBSA is able to achieve the global optimum value in this example.

7.5.3 The Ability of the Market Based Scheduling Algorithm to Handle Rush Orders

Order 3 arrives just before $t = 2T$ and it has a deadline at the end of slot 3. The simulation shows that order 2 was scheduled at slot 3 prior to the arrival of order 3. When order 3 arrives, it is able to grab slot 3 and the order 2 is successfully rescheduled to another slot. Both orders are able to be processed before their deadlines. This example shows that the MBSA is able to handle rush orders.

In a conventional scheduling algorithm, when a rush order arrives and rescheduling is required, the entire scheduling algorithm is repeated with the added new information. However, the MBSA does not required re-evaluation of the entire schedule. The algorithm simply works by having the new order bidding for slots before its deadline and the outbid orders that lose their slots look for other slots.

Here, it is noted that an order that comes late will only be able to get its required slots if it has sufficient value to bid for the slots. In the case where it has a low value, it will not be able to compete with the existing orders and might not get slots before its deadline. This corresponds to a low priority order that has an early deadline. In such situation, the solution that does not complete that order may be better.

7.5.4 Verification of Simulation Results by Comparing All Slots are Packed within Their Slots

We see in Figure 20 on page 65 and in Figure 21 on page 66 that all jobs are processed in their respective slots. We verified that the results obtained in the simulations are a good representation of that on an actual physical system. There is no job that exceeds its slot and there is no job that misses its slot. This is due to the selection of D which enables all jobs to be available to be processed in their respective slots and the selection of T which guarantees that all jobs finishes before the end of their slots.

7.5.5 The Difference between One Docking Station and Two Docking Stations

Comparing Q 4 on page 53 and Q 9 on page 62, we see that for a value of D, Q for a system with one docking is greater than that with two docking stations. This means that for a given D, the maximum number of shuttles in a side loop for one docking station is greater than that for two docking stations. The physical configuration of the side loop may restrict the total shuttles in the side loop. So, by using a larger number of docking stations, the feasible range of D increases.

For a given D, Q is smaller when two docking stations are in use. As $T_{operation}(Q)$ normally increase monotonically with Q, a smaller T may be selected. A smaller T means that each slot has a smaller period and more slots can be scheduled for a given period of time.

Summarising the points above, a system with a more resources (for example, a larger number of docking stations in the Auto-ID Lab setting) benefits from the following. Some of them are dependent on the others.

1. Less products clutter at the vicinity of resources.
2. A larger value of D is possible.
3. A smaller slot period is possible.

7.6 Effect of Varying Slot Period on the Global Value

In the previous section, T and D are calculated such that even if all shuttles take T_{main_loop} to travel from the main loop into the side loop, take $T_{side\_loop}(Q)$ to travel from the side loop into the docking station and $T_{packing}$ to be packed, all boxes can still be packed within their slots. However since, most of the time, the actual time taken for these operations are less than the maximum values, we could expect the jobs can still be carried out within their slots even if a smaller T is selected.

7.6.1 Setup

We consider a system with four orders, each requires one slot. The calculations of the number of slots, the reserved prices, the orders' value and D are shown in Appendix E. We simulated a range of T from 12s to 58s. In each simulation, if an order cannot be packed in its slot, it bids again for another slot.

7.6.2 Simulation Results

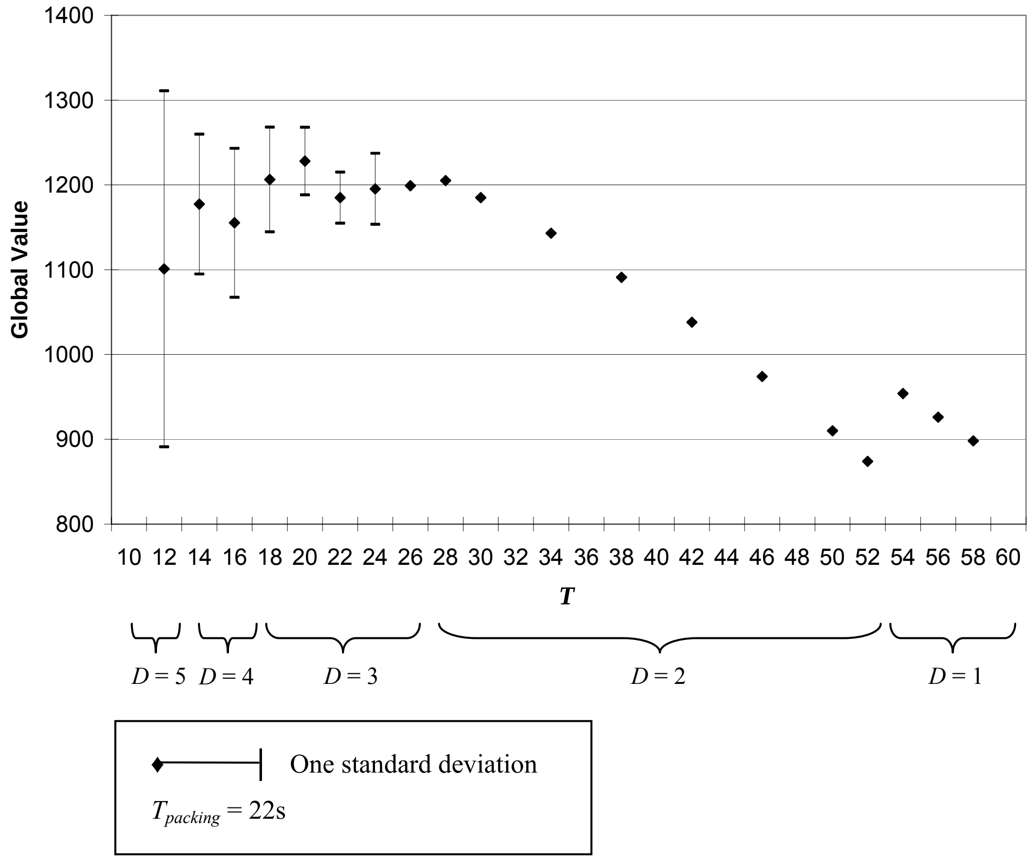


Figure 22. A graph showing the effect of varying T on the global value.

Figure 22 shows the averaged global value of solutions by the MSBA when T changes. For each T, the simulations are repeated ten times, except for $T = 12\text{s}$ where twenty simulations are carried out. The global value of the solution in each simulation may vary as the simulated travelling time for shuttle and packing time are different in each simulation.

7.6.3 Discussions

From the graph, it is noted that:

1. As T decreases, the global value increases. This is because if a job can be completed within its slot, selecting a smaller T frees up more time and thus increases the global value.
2. There is a dip in global value when D increases. This is because the slots are in monotonically increasing reserved prices (shown in Appendix E). When D increases, a job has to select a later slot, which is more expensive.
3. When the slot period approaches $T_{packing}$, the global value drops as some jobs miss their slots and they have to re-bid for other slots. This decreases the global value.
4. For $T \geq 26\text{s}$, the standard deviation of the global value is 0. This indicates that no job misses its slot and thus every simulation produces the same global value. However, when $T < 26\text{s}$, some jobs miss their slots and they have to re-bid for other slots. The variance gets larger as T gets smaller, because the probability for a job not being able to be completed within its slot is higher. When the job cannot be completed within its slot, it bids for another slot and it may face the same problem of not being able to be completed in the new slot.

CHAPTER 8 CONCLUSIONS

This research investigates the use of a MBSA in a distributed manufacturing system. We performed simulations and analyses on the performance of the algorithm and successfully implemented the algorithm in Cambridge Auto-ID Lab, an HMS testbed.

8.1 Original Contributions

Our original contributions are:

1. We compared the performance of the MBSA under varying number of resources, number of jobs and bidders' bidding strategies. We showed that the algorithm is able to produces close to optimum solutions.
2. We derived the upper bounds of the suboptimality of the MBSA in a system with two jobs.
3. We partitioned the scheduling parameter space in to regions in which the MBSA either (a) always produces optimum solutions, (b) never produce optimum solutions, or (c) sometimes produces optimum solutions. As a scheduling system designer, one may design the system such that the possibility of an optimum solution being produced by the algorithm is higher.
4. We proposed modifications to the MBSA so that it can operate in a dynamic system where jobs arrive randomly and perform rescheduling when rush orders arrive.
5. We implemented the MBSA in an HMS. A scheduling program is written to interface with the existing HMS in Cambridge Auto-ID Lab and to test the algorithm in an HMS.

8.2 Recommendations for Future Research

From the results of this research, we see that the MBSA is a suitable scheduling algorithm to be implemented in a heterarchical, distributed manufacturing system. We derived the boundaries for partitioning the parameter space (reserved prices and bidders' values) in a scheduling problem in which solutions with different optimality may result. However, the analysis is limited to a system with two bidders. We propose a future research to derive a generalised model for a system with any number of bidders.

In this research, we modified the MBSA and implemented it in a real time, dynamic system with (a) one single resource, and (b) two resources with constraints. We recommend future researches to look into multiple resources with different constraints.

8.3 Conclusions

From simulations, we found that the MBSA is able to produce close-to-optimum solutions. However, the solutions degrade when the resource to job ratio decreases. We showed that for a scheduling system with four slots and two jobs, the worst global value produced by the MBSA is at most $q_{\max}$ (which is the largest reserved price of the slots) lower than the optimum global value. We also showed that if the reserved prices of the four cheapest slots are made equal, the MBSA will always produce optimum solutions.

From the implementation of the MBSA in a simulated environment and in a physical HMS system, we found that with our proposed modifications, the MBSA can operate in a dynamic system whereby orders arrive randomly and is able to perform rescheduling to handle rush orders.

APPENDIX A AN EXAMPLE AUCTION SETTING TO SHOW SUBOPTIMALITY OF THE MARKET BASED ALGORITHM

Consider the following auction settings:

1. There are four slots, with reserved prices \$10, \$10, \$8 and \$2.
2. There are two bidders, bidder 0 and bidder 1, with values \$16 and \$17 respectively.
3. Each bidder needs two slots.
4. The price for a slot increases by \$1 after each bid.

By inspection, it is easily seen that the optimum solution is achieved by assigning slots 2 and 3 to bidder 1, leaving slots 0 and 1 unassigned. The global optimum value is \$37.

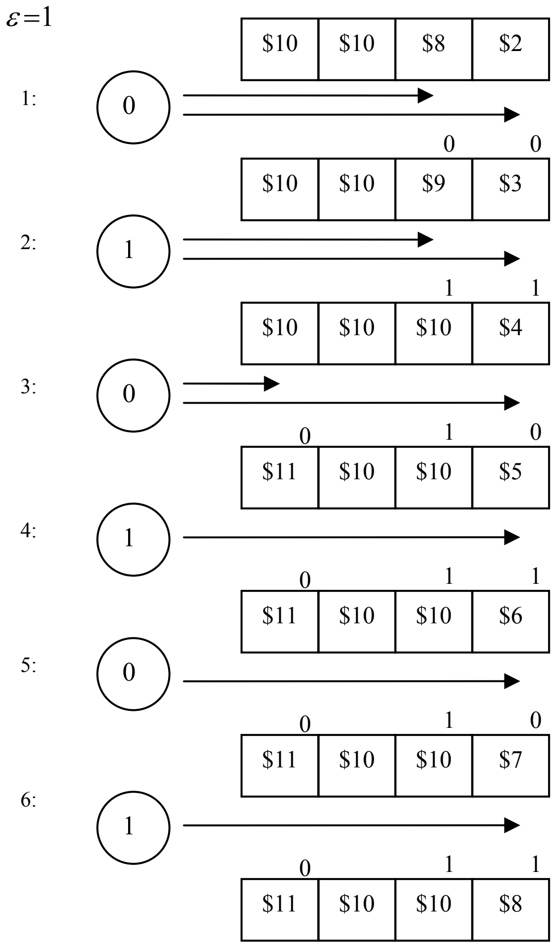


Figure 23. An example showing the MBSA produces a suboptimum solution, with bidders using the SFB strategy.

Figure 23 shows an example of an auction with the above settings. The bidders uses the SFB strategy and the solution achieve by the MBSA has a global value of only $27. Starting from the reserved prices of slots $10, $10, $8, and, $2, bidder 0 bids for slots 2 and 3 because they are the cheapest. Now, the bidder 0 is winning 2 slots and the prices for the slots is now $10, $10, $9, and, $3. As the current prices for slots 2 and 3 are still cheaper, bidder 1 will bid for them. The price for slots 0, 1, and 2 are now equal. This is the point when the MBSA goes off the optimum solution. In the optimum solution, slots 0 and 1 are unallocated. However, in the MBSA, the bidders start to bid for them when the price of slots 0, 1, and 2 is equally cheap. As indicated in the third bidding round, bidder 0 bids for slot 0 and slot 3. The bidding continues until round 6 when bidder 1 bids for slot 3 at \$7. It is noted that bidder 1 obtained slot 2 at \$9 in the second bidding round and thus it has sufficient value to bid for slot 3 at \$7 in the sixth bidding round. This makes the prices \$11, \$10, \$10, and, \$8; and the auction terminates as bidder 1 acquires enough slots and bidder 0 has not enough value to bid for any more slots. The global value for this solution is the sum of the reserved price of slot 1 (\$10) and the value of bidder 1 (\$17), which is \$27.

APPENDIX B DERIVATION OF THE UPPER BOUNDS OF THE SUBOPTIMALITY OF SOLUTIONS PRODUCED BY THE MARKET BASED SCHEDULING ALGORITHM

The following system is considered:

1. There are two bidders, bidder 0 and bidder 1, with values v_0 and v_1 respectively. In addition, $v_0 \leq v_1$. Bidder 1 gets a value higher or same as bidder 0.
2. Each bidder requires two slots.
3. Each bidder uses the SFBSCA strategy.
4. There are four available time slots: slot 0, slot 1, slot 2 and slot 3, each with a reserved price of $q_0, q_1, q_2, \text{and}\, q_3$ respectively. Without loss of generality, the slots are in the order of non-increasing reserved price, that is $q_0 \geq q_1 \geq q_2 \geq q_3$. The lowest reserved value is $q_{\min} = q_3$ and the highest reserved value is $q_{\max} = q_0$.

In deriving the suboptimality of a solution, we first break up the problem domain described above into different categories. The categories are divided according to the optimum solution of the problem. After that, we look into each category to find out how suboptimality may arise. This is done by comparing the optimum solution with any suboptimum solutions that may be generated using the MBSA. There are three categories of optimum solutions:

1. The optimum solution $\mathbf{f}_1$ is $[-1\ -1\ -1 -1]$. Under this situation, it is easily seen that $q_i + q_j > v_k$, where$i \neq j, \forall i, j \in \{0,1,2,3\}and\forall k \in \{0,1\}$. In other words, the best allocation is such that no bidder gets any slot. This happen when the sum of the reserved price of any two slots is higher than the value of any bidder. Therefore, no bidder will ever bid for the slots and thus the MSBA will always produce optimum solutions.
2. The optimum solution $\mathbf{f}_2$ is $[1\ 1\ 0\ 0]$ or any permutation. The global value of solution $\mathbf{f}_2$ is $v(\mathbf{f}_2) = v_0 + v_1$. This means the value v_0 and v_1 are both greater than the sum of any two of the reserved prices, particularly $v_0 \geq q_i + q_j$, $\forall i, j$. This is because if any of two reserved prices sums to be greater than v_0, a better solution would have been leaving the two slots unallocated. In other words, in any auction there will be no slot which any bidder is unable to bid. Hence any solution of the MBSA that is different from the permutation of $\mathbf{f}_2$ is not possible since both the bidders will never leave any slots unbid. For example, suppose a suboptimum solution produced by the MBSA is $\mathbf{f}_3 = [1\ 1\ 0\ -1]$ (bidder 1 gets 2 slots, bidder 0 gets 1 slot and slot 3 is unallocated). Here, the value of the solution $\mathbf{f}_3$ must be less than that of $\mathbf{f}_2$ (by the definition of suboptimality). That is $v(\mathbf{f}_2) = v_0 + v_1 > v(\mathbf{f}_3) = v_1 + q_3$ or $v_0 > q_3$. This is a contradiction because bidder 0 will have sufficient value to bid for time slot 3 in the auction. So, $\mathbf{f}_3$ cannot be a solution produced by the MBSA if $\mathbf{f}_2$ is the optimum solution.
3. When the optimum solution is $\mathbf{f}_4 = [-1\ -1\ 1\ 1]$, four possible suboptimum solutions are:
   a. $\mathbf{f}_5 = [-1\ 0\ 1\ 1]$ or any permutation

b. $\mathbf{f}_6 = [1\ 1\ 0\ 0]$ or any permutation

c. $\mathbf{f}_7 = [-1\ -1\ 0\ 0]$ or any permutation

d. $\mathbf{f}_8 = [-1\ 0\ 0\ 1]$ or any permutation (shown to be impossible later)

It is noted that any permutation of $[-1\ 1\ 0\ 0]$ or $[-1\ 0\ 1\ 1]$ cannot be an optimum solution because the global value of $[-1\ -1\ 0\ 0]$ or $[-1\ -1\ 1\ 1]$ is higher. Similarly, any permutation of $[-1\ -1\ -1\ 0]$ or $[-1\ -1\ -1\ 0]$ cannot be an optimum solution because the global value of $[-1\ -1\ -1\ -1]$ is higher.

Any permutation of $[-1\ -1\ 0\ 0]$ is not a global solution when $v_1 > v_0$ because $[-1\ -1\ 1\ 1]$ has a higher global value. When $v_1 = v_0$, the analysis is the same as that in (3) but with extra constraint.

Therefore, only case (3) may results in suboptimum solutions. In order to find the lower bound of the global value, the following cases are considered.

For 3(a), the suboptimum solution is $\mathbf{f}_5 = [-1\ 0\ 1\ 1]$ or any permutation. An auction leads to a solution $\mathbf{f}_5$ when bidder 0 places its bids but subsequently one of the slots is outbid by bidder 1. Furthermore, bidder 0 does not have sufficient value to bid for any other slots. The suboptimality is therefore $v(\mathbf{f}_4) - v(\mathbf{f}_5)$. This value is has the maximum value of $\max[v(\mathbf{f}_4) - v(\mathbf{f}_5)] = \max[(q_0 + q_1 + v_1) - (q_0 + v_1)] = \max(q_1)$. Since $q_0 \geq q_1$, the function is maximised when $q_1 = q_{\max}$.

For 3(b), the suboptimum solutions is $\mathbf{f}_6 = [1\ 1\ 0\ 0]$ or any permutations. The maximum suboptimality is

$\max[\mathrm{v}(\mathbf{f}_4) - v(\mathbf{f}_6)] = \max[(q_0 + q_1 + v_1) - (v_0 + v_1)] = \max(q_0 + q_1 - v_0)$ .The function is maximised by setting $q_0 = q_1 = q_{\max}$. To maximise the suboptimality, v_0 is chosen to be as small as possible. However, there are further constraints on v_0 as described below.

1. Since the reserved prices are arranged in a non-ascending order, bidders will start bidding from slot 2 and slot 3.
2. We see in $\mathbf{f}_6$ that both slot 0 and slot 1 have being bid for, the prices for slot 2 and slot 3 must have reached q_1 before that can happen. When the prices for slot 2 or slot 3 reaches q_1, a bidder may choose to bid for slot 1 (rather than slot 2 or slot 3) since they are equally cheap.
3. From the condition $q_1 = q_{\max}$, bidder 0 must have bid slot 2 at the price $(q_{\max} - \varepsilon)$, making the price of slot 2 to be q_1. This gives bidder 1 the possibility to bid for either slot 0, slot 1 or slot 2.
4. Bidder 1 must be bidding two slots at that time because prior to this stage, both bidders were only been bidding for slot 2 and slot 3. So in each round, both slots won by the same bidder.
5. To minimise v_0 it is set to be only sufficient for bidder 0 to bid the cheapest slot (which is slot 3) at the lowest possible price (which is $q_{\min}$) and also slot 2 at $(q_{\max} - \varepsilon)$ as mentioned in (3) above. Hence the smallest possible value of v_0 is $(q_{\max} + q_{\min} - \varepsilon)$. Hence the maximum suboptimality is $(q_0 + q_1 - v_0) = [(q_{\max} + q_{\max} - (q_{\max} + q_{\min} - \varepsilon)] = (q_{\max} - q_{\min} + \varepsilon)$.

For 3(c), the suboptimum solution is $\mathbf{f}_7 = [-1\ -1\ 0\ 0]$ or any permutation. An auction will terminate with a solution $\mathbf{f}_7$ if bidder 1 is unable to offer higher bids to those slots held by bidder 0 and also unable to bid for the unallocated slots. For $\mathbf{f}_7$ to be a suboptimum solution, the value of bidder 0 must be smaller than the value of bidder 1, that is $v_1 > v_0$. This results in $v(\mathbf{f}_4) > v(\mathbf{f}_7)$. Here, bidder 0 is able to bid the slot 2 and slot 3 at prices p_2 and p_3 respectively, hence $v_0 \geq p_2 + p_3$. After that, the prices of slot 2 and slot 3 are $(p_2 + \varepsilon)$ and $(p_3 + \varepsilon)$ respectively. Since the solution is allocating slot 2 and slot 3 to bidder 0, bidder 1 must have had not enough value to bid, that is $v_1 < p_2 + \varepsilon + p_3 + \varepsilon$. Combining this with the condition in the previous paragraph, we have $v_1 - v_0 < 2\varepsilon$ and the suboptimality is given by $v(\mathbf{f}_4) - v(\mathbf{f}_7) = v_1 - v_0 < 2\varepsilon$.

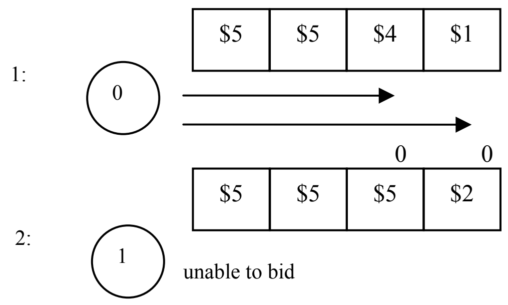


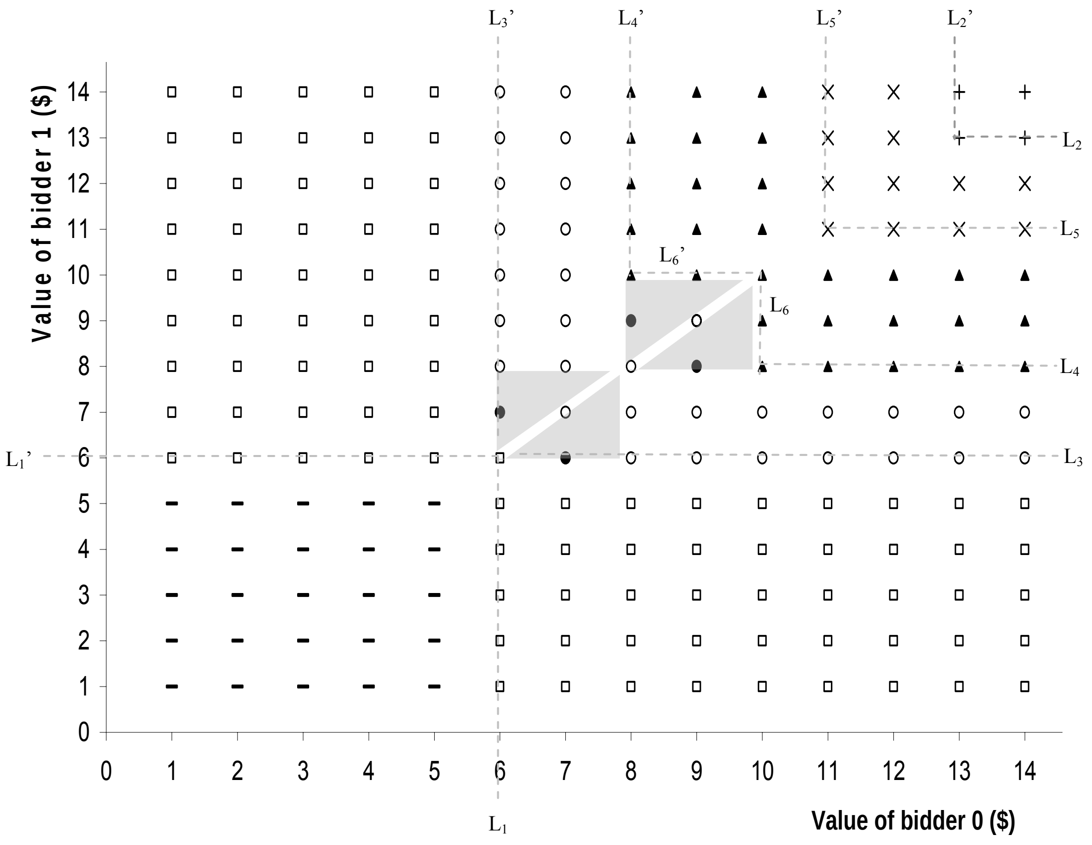
Figure 24. A bidder has a higher value but loses in an auction

An example is given in Figure 24. Here bidder 0 has \$5 and bidder 1 has \$6. The optimum solution is assigning slot 2 and slot 3 to bidder 1 and leaving slot 0 and slot 1 unallocated. The global optimum value is \$16. In the example, bidder 0 bids first. After one round of bidding, bidder 1 is unable to bid because the sum of price of any two slots exceeds its value. The global value of the solution is \$15.

For 3(d), the suboptimum solutions is $\mathbf{f}_8 = [-1\ 0\ 0\ 1]$ or any permutation. As bidder 1 will never bid for only one slot, it must have bid for two slots but one of them is outbid by bidder 0. Hence one of the slot that bidder 0 holds must have a price of $p_i = q_i + k\varepsilon$ before bidder 0 bid for that slot, where i $(1 \text{ or } 2)$ is the slot index and $k \in Z^+$. In the solution, we see that bidder 0 wins slot 1 and slot 2, hence $v_0 \geq p_1 + p_2$ (prices shown are at the time when bidder 0 bids). Bidder 1 is unable to bid further only if $v_1 < p_1 + \varepsilon$ and $v_1 < p_2 + \varepsilon$. But one of these must be false as $v_1 \geq v_0 \geq p_1 + p_2$ and $p_i = q_i + k\varepsilon$. There is a contradiction and therefore $\mathbf{f}_8$ can never result from the MBSA.

APPENDIX C DERIVATIONS OF RESERVED PRICES AND BIDDERS' VALUES THAT PRODUCES SUBOPTIMUM SOLUTIONS

C.1 Specific Set of Reserved Prices



● ▲	The market based algorithm may/may not produce an optimum solution
×	The market based algorithm will never produce an optimum solution
— □ ○ +	The market based algorithm will always produce an optimum solution

Figure 25. A graph showing the effect of bidders' values on the optimality of the solution.

In this section, we investigate the effect of bidders' values on the suboptimality of a schedule. A system consisting of four slots and two bidders is considered. Each bidder requires two slots. The reserved prices of the slots are chosen to be \$7, \$6, \$4, and \$2 for slot 0, slot 1, slot 2, and slot 3 respectively. The values are chosen in such a way that different scenarios in an auction can be demonstrated. Using this example, we show how, by varying the value of the bidders, different solutions are produced by using the MBSA. Figure 25 shows results obtained by analysis. Based on these results, the MBSA will either:

1. always produce optimum solutions,
2. never produce optimum solutions, or
3. sometimes produce optimum solutions depending the sequence of bidding and the slot selection when more than two slots are of equal prices.

If each bidder has less than \$6, none of them will be able to bid for any two slots. Hence the market algorithm produces a solution that assigns no slot to any bidder. We can also see that there is no schedule that gives a higher global value than this. Hence the MBSA will always produce optimum solutions.

On the other extreme end, if both bidders have \$13 or more, they will be able to get 2 slots each. As the sum of the reserved price of any two slots is \$13 or less, the optimum schedule is to assign all four slots to both bidders. If one bidder has exactly \$13, a schedule that assigns slot 0 and slot 1 to this bidder will give the same global value as the schedule that leaves slots 0 and 1 unassigned. However, in both cases, the solutions are optimum. So, the MBSA, which assigns all four slots to both bidders, will definitely achieve optimum solutions when both bidders have \$13 or more.

For any bidders' values other than those two conditions mentioned above, the optimum solution is to assign slot 2 and slot 3 to the bidder with higher value and to leave slot 0 and slot 1 unassigned. To investigate the solutions produced by the MBSA, we look at how the bidders bid and when the auction terminates.

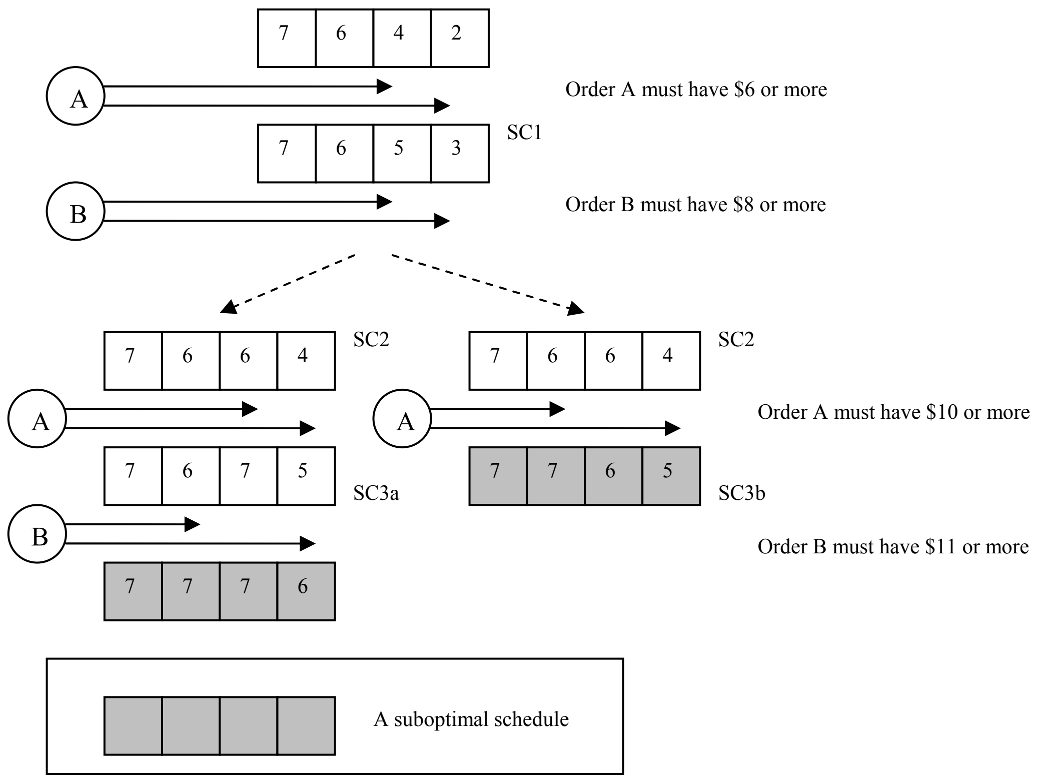

7 6 4 2
A
Order A must have $6 or more
SC1
7 6 5 3
B
Order B must have $8 or more
SC2
7 6 6 4
SC2
7 6 6 4
A
A
Order A must have $10 or more
7 6 7 5
SC3a
7 7 6 5
SC3b
B
Order B must have $11 or more
7 7 7 6
A suboptimal schedule

Figure 26. A bidding process that shows how suboptimality arises.

Figure 26 shows the bidding process of two bidders in an auction. We define bidder A as the first bidder to bid.

Consider the situation when bidder A has $6 or more and bidder B has less than $6. Only bidder A bids and the auction terminates at schedule SC1 (refer to Figure 26). Slot 2 and slot 3 will always be assigned to bidder A. This is marked as □ in Figure 25. The other bidder cannot bid at all as it has not enough value. Under this condition, the MBSA always produces optimum solutions.

If each bidder has \$6 or more, both will be able to bid. Also, if they have less than \$8, no bidder will be able to bid at SC1 in Figure 26. Since at SC1, only the two cheapest slots are being bid for, a solution is optimum when the bidder with higher value gets the slots. These cases are indicated by ○ in Figure 25.

There are cases when the bidder with lower value gets the slots. This happens when after the bidder with lower value bids, the price of the slots exceeds the value of another bidder. For example, each bidder has \$6 or more but less than \$8; or each bidder has \$8 or more but less than \$10. For the former case, any bidder is capable of bidding in the first round. But as each has less than \$8, the bidding will stop at SC1. If bidder A (bidder A will always bid first by definition) has \$6 and bidder B has \$7, bidder A will bid up the prices for slot 2 to \$5 and slot 3 to \$3. Although bidder B has higher value, it will not be able to win the slots. This is a suboptimum solution. On the other hand, if bidder A has \$7 and bidder B has \$6, an optimum schedule can be achieved. Under this condition, we see that the sequence of bidding determines if an optimum solution can be achieved. Hence the MBSA may or may not yield an optimum solution. This is shown as the shaded area (excluding the line $v_0 = v_1$) in Figure 25.

If one bidder has \$8 to \$10 and another has \$10 or more (marked as ▲ in Figure 25), the bidding will reach the stage SC2. At SC2, suboptimality may occur as the price of slot 2 now equals to the price of slot 1. This “gives” the possibility for a bidder to bid for slot 1 and drives the schedule off its optimum solution. Depending on the sequence of bidding, the auction may terminate at SC2 or SC3.

1. If bidder A has less than \$10, the auction terminates at SC2 and the solution is optimum
2. If bidder A has \$10 or more, it continues to bid at SC2:
    a. If it selects slot 1 and slot 3, the auction terminates at SC3a and the solution is optimum.
    b. If it selects slot 2 and slot 3, the auction terminates at SC3b and the solution is suboptimum.

We see that the MBSA may or may not produce an optimum solution, depending on the sequence of bidding and the random selection of the bidder when two slots are of equal price.

If a bidder has \$11 or more, and another bidder has \$11 or \$12, the MBSA will always produce suboptimum solutions. This is because the auction will not terminate at any stage up to SC3a or SC3b. We see that at SC3a or SC3b, bidder B has not gotten enough slots, so it bids further and produces a suboptimum solution.

C.2 A Generalised Model of 4 or more slots and 2 bidders

Considering a system with four slots or more and two bidders (each requires two slots), a general model for the graph in Figure 25 is derived in this section. We define the four lowest reserved prices of the slots to be q_{A}, q_{B}, q_{C}, and q_{D} where $q_{\mathrm{A}} \le q_{\mathrm{B}} \le q_{\mathrm{C}} \le q_{\mathrm{D}}$.

To derive a generalised model, the positions of the boundaries (L_1, L_2, L_3, L_4, L_5, and L_6) are determined based on the reserved prices of the slots. Let v_1 be the value of bidder 1.

1. When both bidders have value less than $(q_{\mathrm{A}} + q_{\mathrm{B}})$, no bidder can bid and hence the equation for L_1 is $\mathrm{L}_1 : v_1 = q_{\mathrm{A}} + q_{\mathrm{B}}$.
2. If each bidder has a value greater than $(q_{\mathrm{C}} + q_{\mathrm{D}})$, an optimum schedule is to allocate the four cheapest slots to both bidders. So, L_2 is $\mathrm{L}_2 : v_1 = q_{\mathrm{C}} + q_{\mathrm{D}}$.
3. L_3 determines the point at which both bidders are able to bid. Rather than cutting towards the axis as L_1, L_3 extends away from the axis. The equation for L_3 is $\mathrm{L}_3 : v_1 = q_{\mathrm{A}} + q_{\mathrm{B}}$.
4. L_4 and L_6 are the boundaries when the price of slot B increases to be equal or greater than the reserved price of the slot C, q_{C}. Say, in each round of bidding, the slot price increases by ε and the price for slot B takes k bidding rounds to be equal or greater than q_{C}. In other words, k is the smallest integer that satisfies $q_{\mathrm{B}} + k\varepsilon \geq q_{\mathrm{C}}$. Here $k = \left\lceil \frac{q_{\mathrm{C}} - q_{\mathrm{B}}}{\varepsilon} \right\rceil$. At this point a bidder can choose to bid slot C and drive the solution into suboptimality. At the same time the price for slot A will be $(q_{\mathrm{A}} + k\varepsilon)$. So the bidder with lower value must have at least $\{[q_{\mathrm{A}} + (k-1)\varepsilon] + [q_{\mathrm{B}} + (k-1)\varepsilon]\}$ to bid for slot B and slot A. The bidder with higher value must have $[\, q_{\mathrm{C}} + (q_{\mathrm{A}} + k\varepsilon)\,]$ to bid for slot C and slot A. So $\mathrm{L}_4 : v_1 = q_{\mathrm{A}} + q_{\mathrm{B}} + 2(k-1)\varepsilon$ and $\mathrm{L}_6 : v_1 = q_{\mathrm{A}} + q_{\mathrm{C}} + k\varepsilon$.
5. It is also noted that suboptimality only occurs when both bidders are able to bid. Hence the suboptimality occurs in the intersection of the regions above L_3, L_4,, and L_6.
6. L_5 is the boundary when either of the bidders will definitely bid for slot C. This happens when the price of the slot B is more expensive than that of slot C and both bidders are able to bid for slot C. In this situation, any bidding

sequence will definitely produces a suboptimum solution. Suppose after l rounds of bidding, the price of slot B exceeds the reserved price of slot C, that is $q_\mathrm{B} + l\varepsilon > q_\mathrm{C}$ where l is an integer. This can also be written as a floor function, $l = \left\lfloor \frac{q_C - q_B}{\varepsilon} \right\rfloor + 1$. The price of slot A after l rounds of bidding is $(q_\mathrm{A} + l\varepsilon)$. This means that both bidders must have at least $\left[q_\mathrm{C} + (q_\mathrm{A} + l\varepsilon) \right]$ to bid for slots C and A. L_5 is thus L_5: $V = q_\mathrm{A} + q_\mathrm{C} + l\varepsilon$.

7. However, in the region above both L_5 and L_2, a global optimum schedule can be achieved as both bidders will get two slots each. Hence suboptimality only occurs above L_5 but below L_2.

L_1’, L_2’, L_3’, L_4’, L_5’and L_6’ are defined in a similar way. It is noted that in the region marked with ● in Figure 25, suboptimality may occur because of the small difference in the bidders’ values.

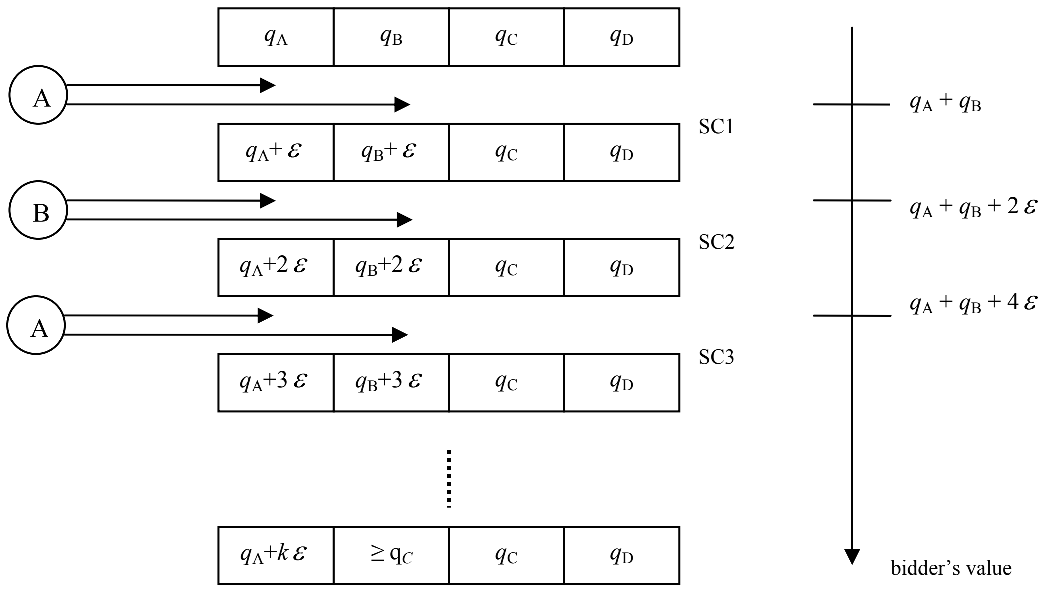


Figure 27. A sequence of bidding to show suboptimality occurs when both the bidders' values fall in certain regions.

Looking at Figure 27, we see that if each bidder has value between $(q_A + q_B) \leq v_i < (q_A + q_B + 2\varepsilon)$ for $i = 1 \text{ or } 2$; the bidding will stop at SC1. The bidder that bids first will win the slot regardless whether its value is higher or lower. In the latter case, a suboptimum solution results. Similarly, if each bidder has value between $(q_A + q_B + 2\varepsilon) \leq v_i < (q_A + q_B + 4\varepsilon)$, the bidder that bids at SC1 wins the slots even if it has lower value. The schedule terminates at SC2 and the other bidder, having less than $(q_A + q_B + 4\varepsilon)$, will not be able to bid further. This trend terminates when the price of slot B exceeds the reserved price of slot C. Beyond this point, suboptimality may occur due to the reason described in (4) above. Summarising, suboptimality may occur if each bidder has value between $(q_A + q_B + 2m\varepsilon) \leq v_i < [q_A + q_B + 2(m+1)\varepsilon]$; until L_6 when a bidder bids slot C instead of slot B. Here $m \in Z_0^+$. When both bidders have equal value, the schedule is always optimum.

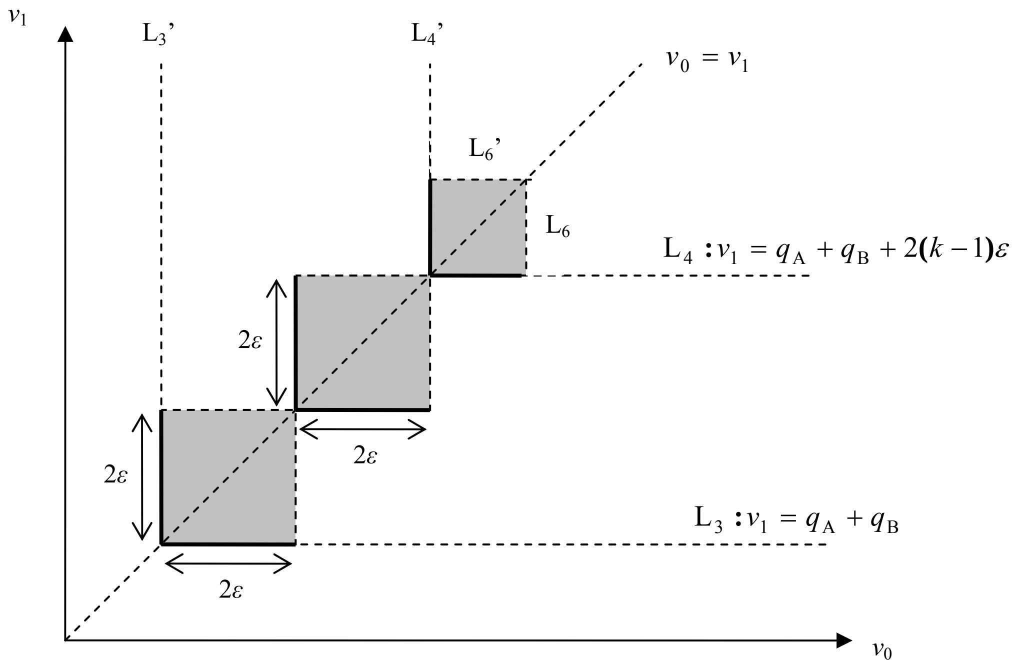


Figure 28. The region when suboptimality occurs due to a small difference between the bidders' values.

The region described above is represented by the grey blocks in Figure 28. The region includes the solid lines but excludes the dotted lines.

APPENDIX D DESCRIPTION OF ROBOT TEST CELL HARNESS

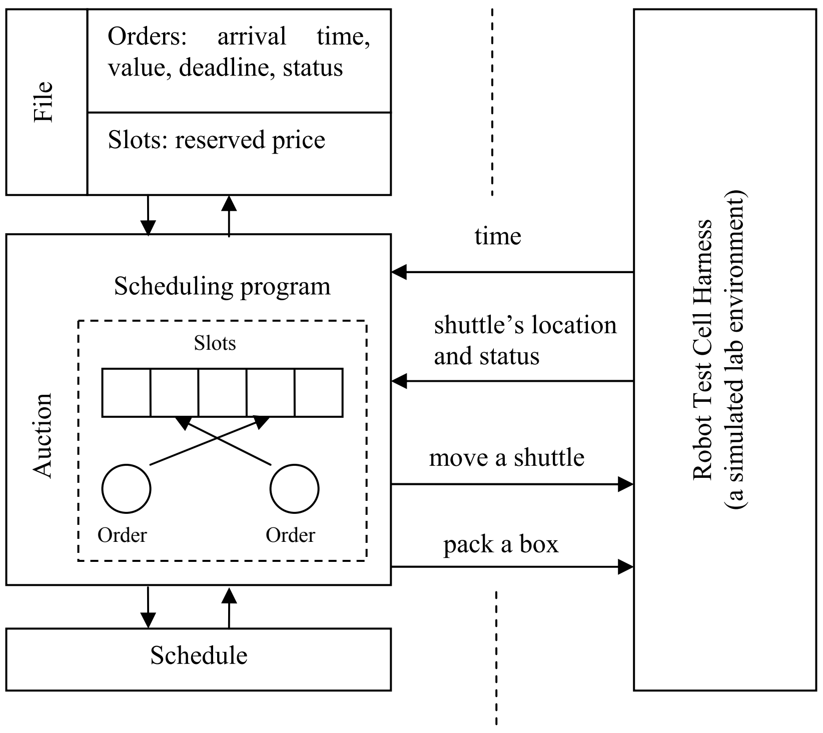


Figure 29. The interface between the scheduling program and the simulated lab environment.

Figure 29 depicts the interface between the program running the MBSA and Robot Test Cell Harness. Robot Test Cell Harness is a software simulation of the robotic cell shown in Figure 15 on page 55. The scheduling program reads a set of orders and slots from a file. The file specifies the time an order arrives, its value, number of time slots that it requires and its deadline. It also specifies the reserved values of the slots. In the scheduling program, the MBSA is executed by simulating auctions where orders bid for slots. An order only "appears" to the scheduling algorithm after it "arrives". The scheduling algorithm continually updates its schedule based on the current winning bidder (which in this case is an order) for each resource (which in this case is a slot).

The scheduling program receives simulated RFID (radio frequency identity) sensor readings, which allow the positions of some shuttles and boxes to be known. The scheduling program sends commands to move a shuttle into the side loop or into the main loop when a shuttle arrives at the gates, to pack a box or to release a box into the side loop when it arrives at the docking station.

APPENDIX E CALCULATIONS OF RESERVED PRICES AND D FOR THE ANALSYIS IN SECTION 7.6

This section shows the calculation for the reserved prices and the value of *D* for the analysis in Section 7.6 on page 70. In the simulation, we consider a system with four orders, each requires one slot. The slots are available between time from 0s to 350s, that is $0 \le t < 350$. For a fair comparison among the configurations of varying slot periods, the reserved price of each slot is determined by the length of the slot and the time of the slot. The price per second at any given time is listed in Table 9. The price per unit time is in an increasing order as we want the jobs to bid for earlier slots. This is to demonstrate that when the jobs are scheduled one after another and the slot period is too small, some jobs may not get to be processed on time.

Table 9. Price per second for the calculation of reserved prices.

Time, t (s)	Price per second ($/s)
$0 \le t < 50$	0.5
$50 \le t < 100$	1.0
$100 \le t < 150$	1.5
$150 \le t < 200$	2.0
$200 \le t < 250$	2.5
$250 \le t < 300$	3.0
$300 \le t < 350$	3.5

For example, a slot period of 50s ($T = 50s$) over $0 \le t < 350$ will produce the following reserved prices:

\$25	\$50	\$75	\$100	\$125	\$150	\$175

Similarly, a slot period of 34s will give the following reserved prices:

\$17	\$26	\$35	\$51	\$61	\$70	\$85	\$96	\$105	\$119	\$35

It is noted that the period of the last slot is less than 34s. This is due to the fact that 350 is not divisible by 34. However, the slot must be included for a fair comparison between different slot period settings. To illustrate this point, consider when no job bids, we end up with a schedule with all unallocated slots. In this case, we want the global value for the schedule to be equal regardless of T. For example, when $T = 50s$, the global value is (\$25 + \$50 + \$75 + \$100 + \$125 + \$150 + \$175 = \$700); when $T = 34s$, the global value is also (\$17 + \$26 + \$35 + \$51 + \$61 + \$70 + \$85 + \$96 + \$105 + \$119 + \$35 = \$700).

In some situations, the last slot is not "complete", that is the slot period is less than T. In those cases, we do not allow any job to bid for that slot although its reserved price is low. Also, in the system, all jobs arrive just before $t = 0\text{s}$ and all of them have a value of \$165. The value is chosen such that all of them are able to get a slot.

D is calculated using the following equation:

$$D = \left\lceil \frac{T_{main\_loop}}{T} \right\rceil = \left\lceil \frac{53}{T} \right\rceil \qquad \text{Q 10}$$

This is to ensure that a shuttle is always available in the side loop at the beginning of its slot.

APPENDIX F CALCULATION OF *Q* FOR A SYSTEM WITH TWO DOCKING STATIONS

This appendix derives the maximum number of shuttles in a side loop for a given value of D when two docking stations are in use.

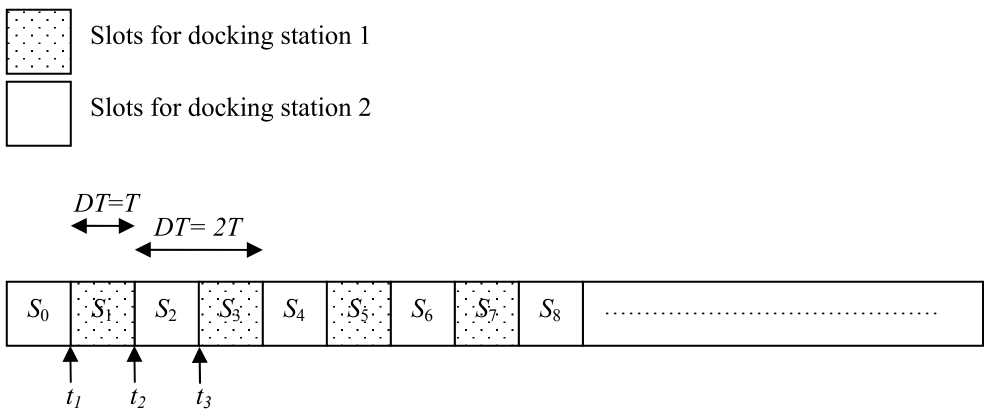


Figure 30. Calculation of Q for a system with two docking stations.

As a robot packing operation will have to be performed within its allocated slot, the slot period T remains unchanged. However, the number of shuttles in each side loop now is different from that when of one docking station is used. Referring to Figure 30, consider only docking station 2:

1. If $D = 1$, the maximum number of shuttles in side loop 2 is two as only shuttles for S_0 and S_2 may be in the loop at $t_1 \le t < t_2$. But at $t_2 \le t < t_3$, only the shuttle for S_2 can be in the loop.
2. If $D = 2$, the maximum number of shuttles in Side Loop 2 is also equal to two. At $t_1 \le t < t_2$, shuttles for S_0 and S_2 can be in the loop. But at $t_2 \le t < t_3$, only the shuttle for S_2 can be in the loop.

3. Consider in the interval of D slots, we see that if D is odd, maximum shuttles happen when the slot before and after D slots belongs to docking station 2. In this case, the number of slots within the D number of slots that belongs to docking station 2 is $\frac{D-1}{2}$. Hence the maximum shuttles in side loop 2, $Q = \frac{D-1}{2} + 2 = \frac{D+3}{2}$.

4. If D is even, the only the slot before or after D slots belongs to docking station 2. There will be exactly $\frac{D}{2}$ slots in any period of D slots that belongs to docking station 2. Hence, $Q = \frac{D}{2} + 1 = \frac{D+2}{2}$.

Summarising the results, the maximum number of shuttles in a side loop at any time is given by:

$$Q = \begin{cases} \dfrac{D+3}{2} & ,\text{if } D = 2n-1 \\ \dfrac{D+2}{2} & ,\text{if } D = 2n \end{cases} \qquad n \in \mathrm{Z}^{+}$$

REFERENCES

[1] Wellman Michael P. , Jeffrey K. MacKie-Mason, Reeves Daniel M., Sowmya Swaminathan, “Exploring Bidding Strategies for Market-Based Scheduling”, EC’03 June 9-12, 2003.

[2] Ferber Jacques, “Multi-Agent Systems”, Addison-Wesley, 1999.

[3] Bongaerts Luc, Valckenaers Paul, Brussel Hendrik Van, Wyns Jo, “Schedule Execution for a Holonic Shop Floor Control System”, ASI 95 of the N.O.E. on Intelligent Control of Integrated Manufacturing System, 24-28/6/1995.

[4] Bussmann Stefan, “A Multi-Agent Approach to Dynamic, Adaptive Scheduling of Material Flow”, Pre-Proceedings, MAAMAW-94, 1994.

[5] Shen Weiming, Norrie Douglas H., “Agent-Based Systems for Intelligent Manufactruing: A State-of-the-Art Survey”, Knowledge and Infomration System, Vol 1(2), p. 129-156, 1999.

[6] Wellman Michael P., Walsh William E., “Auction Protocols for Decentralised Scheduling”, Games and Economic Behavior, Vol. 35, p. 271-303, 2001.

[7] Gu P., Balasubramanian S., Norrie D. H., “Bidding-Based Process Planning and Scheduling in a Multi-Agent System”, Computers ind. Engng, Vol 32(2), pp. 477-496, 1997

[8] Milgrom Paul, “Auctions and Bidding: A Primer”, The Journal of Economic Perspectives, Vol 3(3), p. 3-22, 1989.

[9] Coffman E. G., Bruno J. L., *et al.*, “Computer and Job-Shop Scheduling Theory”, John Wiley & Sons, 1976.

[10] Lynwood A. Johnson, Douglas C. Montgomery, “Operations Research in Production Planning, Scheduling, and Inventory Control”, John Wiley & Sons, 1974.

[11] Andrew S. Tanenbaum, "Distributed Operating Systems", Prentice Hall, Inc., 1995.

[12] "An Application Framework for the Distributed Simulation of Virtual Worlds by Spatial Decomposition". [Online]. Available: http://www.risc.uni-linz.ac.at/people/pkulczyc/dt/text/content.html.

[13] Wooldridge M. J., "An Introduction to MultiAgent Systems", John Wiley & Sons, 2002.

[14] Manoj Kumar, "Internet Auctions", IBM Institute for Advanced Commerce, Technical Paper, 1998. [Online]. Available: http://www.research.ibm.com/iac/tech-paper.html.

[15] McAfee R. Preston, "Auctions and Bidding", Journal of Economics Literature, Vol 25, p. 699-738, 1987.

[16] Wurman Peter R., Wellman Michael P., Walsh William E., "The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents", in Proceedings of the Second International Conference on Autonomous Agents, 1998.

[17] Chirn Jin-Lung, McFarlane Duncan C., "Building Holonic Systems in Today's Factories: A Migration Strategy", Journal of Applied Systems Studies, Vol. 2 (1), 2001.

[18] McFarlane Duncan C., Bussmann Stefen, "Developments in Holonic Production Planning and Control", International Journal of Production Planning and Control, Vol. 11 (6), p. 522-536, 2000.

[19] Gozzi Andrea, Paolucci Massimo, Boccalatte Antonio, "Autonomous Agents Applied to Manufacturing Scheduling Problems: A Negotiation-Based Heuristic Approach", Multi-Agent Systems and Application II, Selected Revised Papers: 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001, LNAI 2322, Springer Verlag, p. 194-203, 2002.

[20] Tharumarajah A., “Survey of Resource Allocation Methods for Distributed Manufacturing Systems”, Production Planning and Control, Vol. 12 (1), p.58-68, 2001.

[21] Milgrom P. R., Weber R. J., “Theory of Auctions and Competitive Bidding, Econometrica, Sept 1982, p. 1089-1122, 1982.

[22] Anthony Patricia, Jennings Nicholas R., “Agents in Auctions: The State of the Art”, Proc. 1st International Conference on Artificial Intelligence in Engineering and Technology (ICAIET-2002), Sabah, Malaysia, p.v725-731, 2002.

[23] Qin Xiao, Jiang Hong, "Dynamic, Reliability-driven Scheduling of Parallel Real-time Jobs in Heterogeneous Systems," in the Proceedings of the 30th International Conference on Parallel Processing (ICPP 2001), pp.113-122, Valencia, Spain, September 3-7, 2001.

[24] Qin Xiao, Han Zongfen, Jin Hai, Pang Liping, and Li Shengli. "Real-time Fault-tolerant Scheduling in Heterogeneous Distributed Systems," in the Proceedings of Cluster Computing Technologies, Environments, and Applications(CC-TEA), the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA, June 26-29, 2000. Vol. I, pp.421-427.

[25] Wooldridge M. J., Ciancarini P., “Agent-Oriented Software Engineering: The State of the Art”, Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing, 2001.

[26] Kraus S., “Automated Negotiation and Decision Making in Multiagent Environments”, Artificial Intelligence Journal, Vol. 104 (1-2), pp. 1-69, 1998.

[27] Jennings N. R., Faratin P., Lomuscio A. R., Parsons S., Sierra C., Wooldridge M. J., “Automated Negotiation: Prospects, Methods and Challenges”, to be appear in Int Journal of Group Decision and Negotiation.

[28] Wellman M. P., “A market-oriented programming environment and its application to distributed multicommodity flow problems”, Journal of Artificial Intelligence Research, Vol. 1, pp. 1-23, 1993.

[29] Wellman M. P., Market-Oriented Programming, http://ai.eecs.umich.edu/people/wellman/MOP.html, June 1997.

[30] Kumar Manoj, “Internet Auctions”, IBM Institute for Advanced Commerce, Technical Paper, 1998. [Online]. Available: http://www.research.ibm.com/iac/tech-paper.html.

[31] Wurman Peter R., Wellman Michael P., Walsh William E., “The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents”, in Proceedings of the Second International Conference on Autonomous Agents, 1998.

[32] Parunak V. D., Baker A., Clark S, “The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design”, Workshop on Agent-Based Manufacturing, ICAA’98, Minneapolis, MN, 10 May 1998, 1998.

[33] Baker Albert D., “A Survey of Factory Control Algorithms that can be Implemented in a Multi-Agent Heterarchy: Dispatching, Scheduling and Pull”, Journal of Manufacturing Systems, Vol. 17 (4), 1998.

[34] Brussel Hendrik Van, Wyns Jo, Valckenaers Paul, Bongaerts Luc, Peeters Patrick, “Reference Architecture for Holonic Manufacturing Systms: PROSA”, Computers in Industry, Vol. 27, p. 255-274, 1998.

[35] Gou Ling, Luh Peter B., Kyoya Yuji, “Holonic Manufacturing Scheduling: Architecture, Cooperation Mechanism, and Implementation”, Computer in Industrial Vol. 37, p. 213-231. 1998.

[36] McEleney B., O'Hare G. M. P., Sampson J, “An Agent Based System for Reducing Changeover Delays in a Job-Shop Factory Environment”, In Proc. of PAAM'98, London, 1998.

[37] Kis Tamas, Vancra Josef, Markus Andas, “Controlling Distributed Manufacturing Systems by a Market Mechanism”, the 12th European Conference on Artificial Intelligence (ECAI), 1996.

[38] Lin Grace Yuh-Jiun, Solberg James J., “Integrated Shop Floor Control Using Autonomous Agents”, IIE Transactions, Vol. 24 (3), 1992.

[39] Bussmann Stefan, “Agent-Oriented Programming of Manufacturing Control Tasks”, Proc. 3rd Int. Conf. on Multi-agent Systems (ICMAS'98), Paris, France, 1998, p. 57-63, 1998.

[40] Chokshi N. N., McFarlane D. C., “Rationales for Holonic Applications in Chemical Process Industries”, Multi-Agent-Systems and Applications II, p. 336-350, 2002.

[41] Walsh William E., Wellman Michael P., Ygge Fredrik, “Combinatorial auctions for supply chain formation”, in ACM Conference on Electronic Commerce, p. 260-269, 2000.

[42] Wurman Peter R., Walsh William E., Wellman Michael P., “Flexible double auctions for electronic commerce: Theory and implementation”, Decision Support Systems, Vol. 24, p. 17-27, 1998.